ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

MINES ParisTech

École doctorale n° 432 : Science des Métiers de l'Ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École nationale supérieure des mines de Paris

Spécialité " Informatique, temps réel, robotique et automatique "

présentée et soutenue publiquement par

Manu ALIBAY

le 18 décembre 2015

Fusion de données capteurs étendue pour applications vidéo embarquées

Extended sensor fusion for embedded video applications

Directeur de thèse : **Philippe FUCHS**
Co-encadrement de la thèse : **Bogdan STANCIULESCU**
Co-encadrement industriel : **Stéphane AUBERGER**

Jury :

M. Gustavo MARRERO CALLICO, Associate professor, Universidad de Las Palmas de Gran Canaria	Rapporteur
M. Mounim YACOUBI, Professeur, Telecom SudParis	Rapporteur
M. Didier AUBER, Directeur de recherche, LEPSIS - IFSTTAR	Président
M. Stéphane AUBERGER, Responsable R&D, STMicroelectronics	Examinateur
M. Bogdan STANCIULESCU, Maître de conférences, MINES ParisTech	Examinateur
M. Philippe FUCHS, Professeur, MINES ParisTech	Examinateur

T H È S E

MINES ParisTech
Centre de Robotique
60 bd Saint Michel, 75006 PARIS, France

Remerciements

Ces trois années de thèse furent un réel plaisir pour moi, grâce à la présence et au soutien de nombreuses personnes, tant sur les plans personnels que professionnels.

Sur le plan personnel je tiens à remercier ma famille qui a été très présente et a suivi de très près mon travail. Je pense notamment à mon papi et ma mamie qui ont toujours montré énormément d'intérêt, allant même jusqu'à mieux connaitre les photos de mon pot que moi ! Merci également à toute la famille du sud-ouest, dont j'ai reçu beaucoup de messages de soutien qui m'ont énormément aidé.

La famille parisienne fut extrêmement présente également, avec de nombreuses attentions et messages. Je remercie mes oncles et tantes avec qui au cours de nombreux diners nous avons bien ri, et je tiens particulièrement à féliciter ma tante Nadine qui a complétement compris le sujet au cours de la soutenance.

Merci à papa et tout le monde de Lanirano, j'espère bientôt retourner vous voir. Vos messages m'ont beaucoup touché.

Un énorme merci à l'équipe et à tous les collègues de ST, avec qui ces trois ans vont se prolonger. Vive le BK et la baballe, ainsi que les blue genies et les portes qui claquent. Je tiens à saluer Arnaud qui s'est battu non seulement pour ce poste en thèse mais aussi pour me garder par la suite.

Cette thèse n'aurait été qu'une fraction de ce qu'elle est sans l'investissement énorme de Stéphane. Tu as été un super encadrant, patient et très attentif à mon travail, je te remercie beaucoup pour tout ce que tu as fait. Bon en termes de rêves j'ai souvent cauchemardé sur les 150 allers-retours pour les présentations et documents, mais c'est dans la tête tout ça.

Merci à Philippe pour sa disponibilité et sa participation, ainsi que Christine pour sa patience et son aide précieuse qui m'a aidé à surmonter ma phobie administrative.

Ce serait mentir de remercier qui que ce soit d'autre pour l'encadrement de ma thèse, je ne le ferai donc pas.

Mes amis et proches ont été supers pendant ces trois ans où on a bien besoin de décompresser. Avec bon nombres d'Adios Motherfucka et de bière-pong. De bons moments passés ensembles, parfois dont je ne me souviens que vaguement ! Merci à toute la bande pour tout, même à ceux qui sont loin en termes de distance seulement.

Je tiens à saluer tous les potes de STMicroelectronics : apprentis, stagiaires, etc… Je tiens aussi à remercier tous les doctorants aux Mines pour ces supers moments passés, notamment dans la neige !

Enfin je dédie ma thèse à maman et doudou, qui ont été là pour me supporter au quotidien (et faire le délicieux pot de thèse). Vous m'avez motivé, soutenu, nourri, râlé dessus pour mes t-shirts et chaussettes troués … mais vous avez aussi su me transmettre beaucoup tous les jours.

Manu

Abstract

This thesis deals with sensor fusion between camera and inertial sensors measurements in order to provide a robust motion estimation algorithm for embedded video applications. The targeted platforms are mainly smartphones and tablets.

We present a real-time, 2D online camera motion estimation algorithm combining inertial and visual measurements. The proposed algorithm extends the preemptive RANSAC motion estimation procedure with inertial sensors data, introducing a dynamic lagrangian hybrid scoring of the motion models, to make the approach adaptive to various image and motion contents. All these improvements are made with little computational cost, keeping the complexity of the algorithm low enough for embedded platforms. The approach is compared with pure inertial and pure visual procedures.

A novel approach to real-time hybrid monocular visual-inertial odometry for embedded platforms is introduced. The interaction between vision and inertial sensors is maximized by performing fusion at multiple levels of the algorithm. Through tests conducted on sequences with ground-truth data specifically acquired, we show that our method outperforms classical hybrid techniques in ego-motion estimation.

Keywords: Motion estimation, computer vision, inertial sensors, sensor fusion, real-time, embedded, RANSAC, SLAM, odometry, particle filter.

Contents

Chapitre 1: Introduction

Below is a French summary of chapter 1: Introduction.

La popularité des téléphones intelligents (ou smartphones) et des tablettes, fait que de nombreuses applications sont développées pour ces plateformes. Une partie de ces applications est en relation avec la vidéo : amélioration de qualité du contenu, reconnaissance de scène ou de personne, réalité augmentée… Une étape clé de toutes ces applications est l'estimation du mouvement de l'appareil. Pour ce faire, plusieurs moyens sont à dispositions sur les téléphones intelligents et tablettes. La vidéo est très souvent utilisée afin d'estimer le mouvement, mais d'autres capteurs sont également exploitables, tels que les capteurs inertiels.

Le but de cette thèse est d'étudier la fusion des données de caméras et de capteurs inertiels afin d'estimer de façon précise et robuste le mouvement de la plateforme embarquée pour des applications vidéos. Ce chapitre introduit les applications concernées, les capteurs inertiels et visuels, les bases de la vision par ordinateur ainsi que les divers modèles de mouvement existants.

Chapter 1: Introduction

With the massive popularity of smartphones and tablets, various applications are developed on those devices. Part of them concern video-related content: improving the quality of the video sequence, computing semantic information on the content of the scene, superimposing contents for augmented reality or gaming… A key step in many of these applications is to estimate the motion of the platform. In order to perform this, one can use directly the camera recordings or other sensors available on the smartphone or tablet, such as the inertial sensors. The goal of this thesis is to perform sensor fusion between the camera and inertial sensors measurements for these embedded video applications.

In section 1.1 of this introductory chapter the two main actors of this thesis are presented: STMicroelectronics and the Robotic Center laboratory from Mines Paristech'. STMicroelectronics has demonstrated over the years many algorithms for embedded video applications using cameras on smartphones. One of the possible improvements of those techniques is to develop fusion methods with inertial sensors. The Robotic Center laboratory displays a high expertise on sensor fusion for mobile robotic, with a wide array of sensors (cameras, laser, GPS, etc…). This knowledge can be transposed to embedded platforms such as smartphone and tablets.

Section 1.2 introduces the main applications that were targeted in the thesis. Firstly, image quality enhancement techniques should be improved with inertial sensors. Robustness improvement and assessment of those need to be performed. Secondly, Simultaneous Localization And Mapping (SLAM) methods should be developed, targeting the current robustness and accuracy issues that concern state of the art techniques.

A deeper look into the two types of sensors studied in the thesis is proposed in section 1.3 . Video sensors, or cameras, possess very complex characteristics, which lead to many possible artifacts with various causes and effects. Inertial sensors on embedded platforms such as Smartphones display high noise values. This leads to the application of ad hoc strategies when performing processing on those types of sensors.

Motion models are discussed in section 1.4 . Being the heart of this thesis, one should carefully define the possible types of models that are encountered when performing motion estimation applications. 2D motion model selection is often dependent on the degree of freedom needed in the desired application. 3D motion model choice is also very variable, but the main concern is a tradeoff between interpretability of the data and good mathematical properties such as the lack of singularities and easiness of differentiability.

Finally, section 1.5 presents the structure of the thesis.

1.1 Context

This thesis was realized in the "Digital Convergence Group" unit of STMicroelectronics. The academic partner is the Robotic center of Mines Paristech. The subject under study concerns mainly the extended fusion between a video sensor (a camera) and inertial sensors (such as gyroscope, accelerometers…) for embedded video applications.

1.1.1 STMicroelectronics

STMicroelectronics (ST) is a worldwide actor on the semi-conductor market, having client covering the range of technologies « Sense & Power » and applications of multimedia convergence. From power management to energy saving, data confidentiality and security, health and care to smart public devices, ST is present where the micro-technology brings a positive and novel contribution to everyday life. ST is in the heart of industrial applications and entertainment at home, desk, mobile devices and cars.

Particularly, in the « Digital Convergence Group », solutions for image and video processing are developed and have been integrated for many years for cellular phones. The "Algorithm for Image Quality" group mission is to design and develop algorithms for those types of processing, specifically for the embedded domain. This includes reference code implementation, platform implementation and inputs for new hardware blocs. The group is also in charge of evaluation activities, standards tracking, coordination and technical expertise for clients or internally. It is also very active for patenting ideas associated to these innovations.

1.1.2 The Robotic Center

One of the major axes of research of the Robotic Center from the "Ecole des Mines de Paris" is to develop tools, algorithms, and applications of real-time analysis from readings coming from multiple sensors, including cameras. Many applications were designed in the domains of virtual reality, augmented reality, driving assisting systems, and video surveillance. Real-time recognition techniques have been developed in this context [Zaklouta & Stanciulescu 2011; Moutarde et al. 2008; Stanciulescu et al. 2009].

A software platform resulting from the center work is today commercialized by the company INTEMPORA under the name of Real-time Advances Prototyping Software (RT-MAPS). It allows the acquisition, prototyping, and execution of real-time processing of synchronized readings of sensors. It is used by many companies and academic research labs such as Thales, Valeo, PSA, Renault, the INRIA, the INRETS, etc… including several projections such as Cybercars[1] and CityMobil[2]. Finally, the laboratory has acquired a solid expertise on the use of

[1] http://www.cybercars.org/
[2] http://www.citymobil-project.eu/

several methods for data fusion (particle filtering, possibilities theory, etc…) in a real-time context.

1.2 Aimed applications

Fusion methods between a video sensor and inertial sensors can lead to or improve many potential applications. The study of this type of fusion in the context of this thesis can be summarized in two parts: video quality improvements, and Simultaneous Localization And Mapping (SLAM).

1.2.1 Video quality improvement methods

Main video applications for embedded platforms are: video stabilization [Im et al. 2006], panorama creation (up to 360 degrees with loop closure) [Brown & Lowe 2006], temporal denoising [Lagendijk & Biemond 1990], or increase of the dynamic by multiple frame alignment [Kang et al. 2003]. Each of those techniques is already under study in ST [Auberger & Miro 2005]. Camera motion estimation is a key step in those applications.

To estimate the camera motion on a smartphone or a tablet, two main sources of information are very relevant: inertial sensors and visual strategies. The first goal of this thesis is to fuse these two. Sensors fusion combined with classical motion estimation techniques should allow an improvement of the robustness of many algorithms based on camera motion. This should remove ambiguities that occur in many purely visual analyses: distinction between local and global motion, immunity to light changes, noise, low textured scenes. Overall, this fusion should lead to an improvement of the vision-based motion estimation without loss in accuracy.

Furthermore, some of these methods make use of extraction of points of interest or keypoints to compute the motion between frames. Inertial data can also produce additional information for a better selection and tracking of these points. This could include prediction of the displacement of points, or the deletion of some constraints of certain descriptors.

1.2.2 SLAM for indoor navigation and augmented reality

The second major part of the thesis concerns the domain of SLAM for 3D mapping and indoor navigation. The estimation of camera motion along with the mapping of a 3D scene from an image sequence is a technique called Simultaneous Localization And Mapping (SLAM). Many studies have been proposed concerning this subject [Davison et al. 2007] [Klein & Murray 2007]. The simultaneous motion estimation and building of the 3D scene is a very challenging problem with the sole utilization of video sequence. In the literature, applications created for mobile robotics or augmented realities encounter robustness issues. Some approaches

combining cameras with other sensors (stereo-vision, laser, GPS) improve the robustness but are not relevant in the context of embedded platforms.

The ROBOTIC CENTER has acquired a certain amount of experience in the fusion of sensors for the SLAM in the DGA-ANR challenge Carotte[3]. The "Corebots" prototype has won the competition twice on three attempts, by localizing itself and mapping in a more accurate manner than its opponents in an outdoor environment. Figure 1-1 shows an example of mapping by the Corebots prototype.

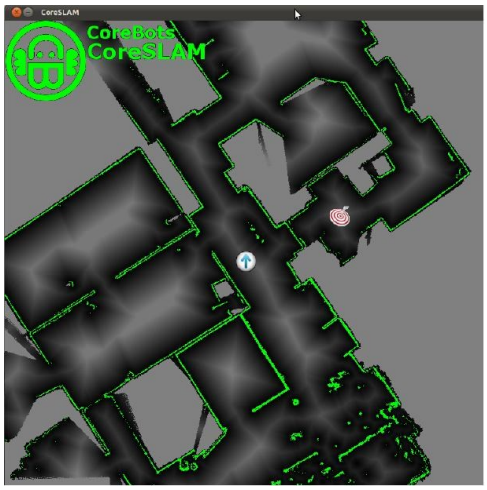


Figure 1-1: Mapping by the Corebots prototype.

Recently, the main drawbacks of state-of-the art SLAM approaches have been the following: heavy computational cost, relatively complex methods, inaccurate results based only on video sensors. This is where the use of fusion with inertial and magnetic sensors should play a decisive role by completing the image and playing as substitution for more complex sensors used in tradition mobile robotics. The estimation of the camera motion is tightly coupled to inertial and visual fusion, allowing the improvement of the 3D trajectory, therefore also impacting the 3D mapping. A decrease of the computational cost is also aimed at as in [Castle et al. 2007].

1.2.3 Main axis of research

Beyond the pure algorithmic improvements, a certain amount of questions should be answered concerning the feasibility and the interest of inertial visual fusion in industrial processing, especially when applied to embedded platforms:

- What are the performance improvements that can be reached compared to the usage of only one sensor (typically pure visual techniques)?

- What are the complexity increases created by the fusion? In terms of direct computational time as well as complexity.

[3] http://www.defi-carotte.fr/index.php

- What is the accuracy needed in terms of calibration? This concerns both temporal calibration (data synchronization) and spatial calibration (axis alignment between the sensors).

The assessment of these points for every developed technique in this thesis is a very important aspect of the work provided. In effect, the industrial viability of a method has to be demonstrated for these algorithms to be relevant for ST.

1.3 Sensors

The main goal of this thesis is to build robust and computationally efficient methods for motion estimation, based on camera and inertial sensors measurements. A quick overview of the respective sensors capabilities and characteristics is given is this section.

1.3.1 Vision sensors

Many possible sensors can be utilized in order to estimate motion for an embedded platform in a video application. Cameras are a very straightforward choice, as the video sequence is the direct target of the application. Many techniques have been developed in order to estimate motion based on the video sequence, and many are already integrated in industrial applications such as video stabilization, panorama creation, tracking, video encoding, etc… Computing the motion from the video sequence alone is not a simple task, especially when the device is handheld, leading to artifacts and difficult cases.

When a device with the camera is handheld, it can undergo high motions (often corresponding to wrist twists). Heavy movements not only require specific steps or considerations in the motion estimation process, but they can also lead to visual artifacts. A commonly known issue is motion blur. As motion occurs while recording a scene during the exposure time, pixels do not record the same place. This implies that nearby pixels will influence the final recording of a pixel, degrading the image quality as illustrated in Figure 1-2.

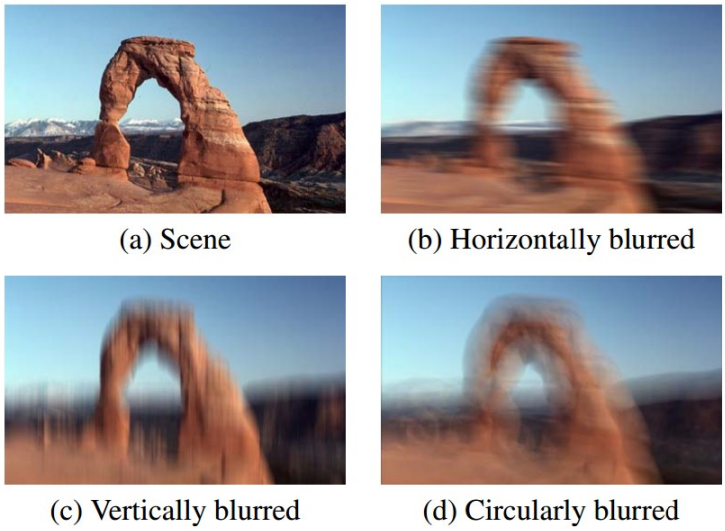


Figure 1-2: Motion blur examples from [Ben-Ezra & Nayar 2003]**.**

Embedded cameras are often based on CMOS (Complementary Metal–Oxide–Semiconductor) transistors. The specificity of this type of cameras is that pixel rows are recorded sequentially rather than simultaneously, as shown on Figure 1-3. This technique is called the Rolling Shutter. A motion occurring between the acquisitions of the lines can lead to distortions in the image, not only degrading image quality but also the motion estimation to a lesser extent. Rolling Shutter distortions can lead to skewed images (Figure 1-3) as well as stretches or compressions.

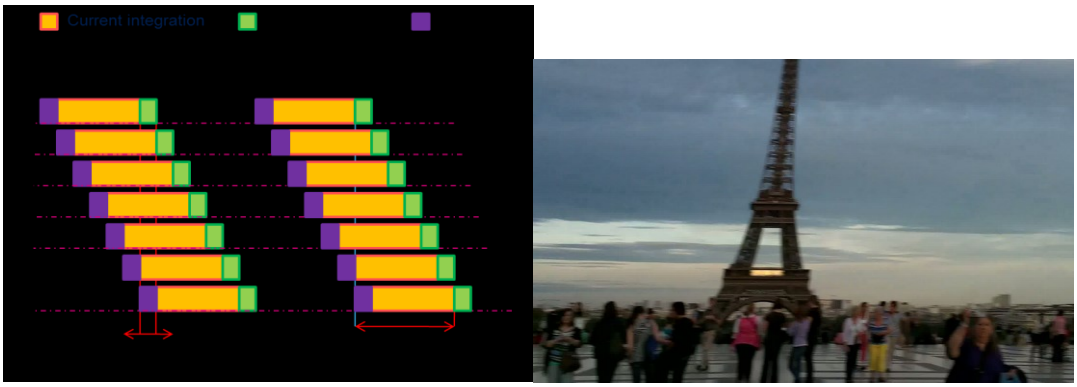


Figure 1-3: Left: Rolling Shutter readout system. Right: Example of skew due to Rolling Shutter distortions.

While these types of effects are often considered as an image quality degradation, they can also affect the motion estimation process because of the geometric changes that they apply on the images. Furthermore, embedded cameras have lower field of view than usual cameras, leading to a lesser angle of the scene recorded, lowering the amount of usable information. They also have a lower resolution, leading to a less detailed content. To conclude, embedded motion estimation induces heavy requirements in terms of robustness to large motions and its consequences in terms of artifacts and distortions.

1.3.2 Inertial sensors

Most recent smartphones and tablets have inertial sensors integrated in their hardware. The three widely used types of inertial sensors are the gyroscope, the accelerometer and the magnetometer. Most embedded inertial sensors are MEMS (Microelectromechanical systems), notably known for their low consumption and volume. An example of MEMS sensors on a board can be seen on Figure 1-4. Other types of sensors are present such as pressure, light and proximity sensors. However, in the context of motion estimation they appeared less relevant. Each type of sensor possesses its own characteristics and they can be used directly or fused to perform a large range of applications. The fusion process consists in performing an estimation using measurements from multiple sensors types. This allows to overcome specific drawbacks from each sensor, even if some remain after using fusion techniques.

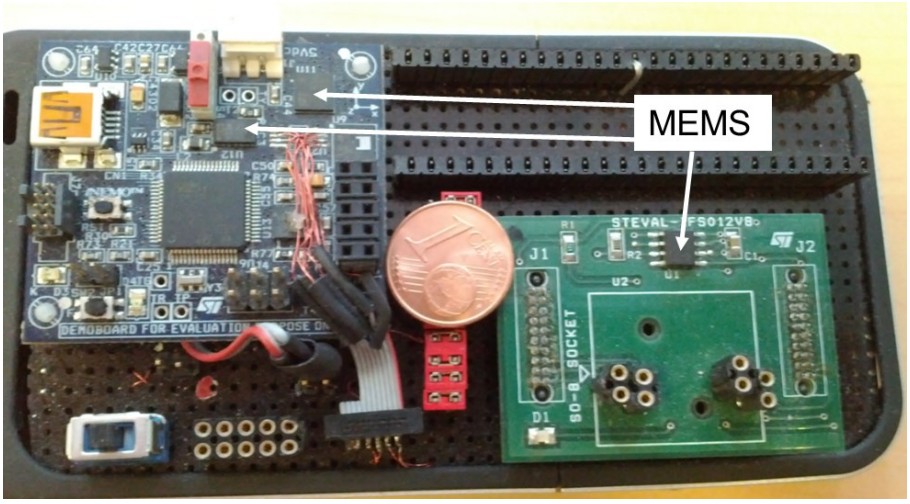


Figure 1-4: MEMs sensors on a board, with a 1 cent of Euro piece for scale.

1.3.2.1 Sensors characteristics

The gyroscope measures rotation speed in three dimensions. The measurement rate ranges from 50Hz up to 500Hz. To obtain orientation information, it is necessary to integrate the data from the gyroscope, giving only a relative estimation on the rotation of the device from one measurement to the next. It leads to error accumulation, also known as drift, when measuring a biased sensor, which is always the case for MEMS gyroscopes. In our experiments, we noticed this type of sensor was the least noisy of the three, but as it only gives a relative estimation and it can drift, either fusion or specific filtering needs to be performed before using its measurements.

The accelerometer indicates the mobile acceleration that is composed of the gravity and the mobile's own acceleration. In the MEMS case, the sensor is usually very noisy. It is therefore very challenging to retrieve the true mobile acceleration from it, and even harder to estimate the speed or the position, as it requires integration of a highly noisy signal. Therefore the accelerometer is usually only used to deduce the direction of the gravity, providing an estimation of the mobile orientation without any integration needed. It must be considered that

making this assumption implies that the mobile's own acceleration then becomes a form of noise.

The magnetometer estimates the earth magnetic field direction and so the magnetic north. Similarly to the accelerometer, the measures are very noisy. In addition, magnetic perturbations add to the noise; the effects known as "soft iron" and "hard iron" lead to distortion in the local magnetic field. The hard iron is due to a magnetic reflection from an object, while the soft iron comes from interferences in the magnetic field, for instance due to a cellphone. Figure 1-5 displays an example of what a magnetometer measures when performing a 360° yaw rotation. We only represent the x and y axis, as the vertical axis does not vary. The upper graph shows the measurement without perturbations. The lower-left one displays the hard iron effect, and the lower-right one the effects of the soft iron. It can be seen that without any disturbances the north direction describes a round circle while the 360° turn is performed. However, in the presence of either soft or hard iron, the circle is either shifted (hard iron) or not round anymore (soft iron), which induces errors when estimating the north direction.

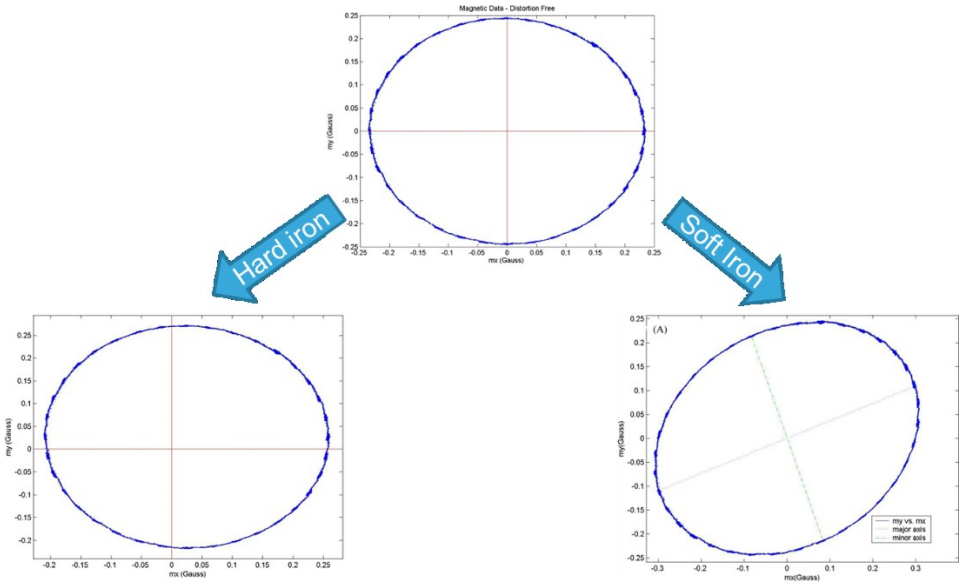


Figure 1-5: The effect of soft and hard iron on a magnetometer when performing a 360° turn.

To sum up, each sensor has its own issues:

- The gyroscope provides a rotation speed estimation. Performing orientation estimation requires integration, which can create drift from biases.
- The accelerometer is very noisy, and as we want to estimate the gravity direction only, platform's acceleration is to be added to noise.
- The magnetometer is also very noisy, and can undergo a lot of distortions due to various magnetic effects.

It is necessary to overcome those drawbacks by fusing the sensors to obtain a robust and accurate estimation of the orientation of the mobile.

1.3.2.2 Inertial sensors fusion techniques and applications

There are a lot of applications domains for inertial sensors fusion in embedded platforms, from Unmanned Aerial Vehicles (UAVs) navigation to Smartphone applications. For the Smartphones and Tablets the targets are mainly gaming, orientation awareness, pedestrian step counting, etc… The goal of inertial sensor fusion is to utilize the measurements from the three sensors simultaneously in order to compute the 3D orientation (and eventually position, even if it is highly challenging for low-cost sensors [Baldwin et al. 2007]) of the device. Two main techniques are utilized in the literature to accomplish this: the Kalman filter and the complementary filter.

The **Kalman filter** [Kalman 1960] is utilized in many domains and applications. It estimates over time a set of Gaussian unknown variables, called the state of the filter. The first two statistic moments are computed, that is the mean and the covariance of the state. This estimation is based on a physical model and measurement equations. Like most Bayesian filters, it proceeds in two steps: a prediction of the state of the filter and a correction of the prediction based on the measurements. The detailed operation of the Kalman filter is shown in Appendix A: The main requirement to apply it for inertial sensor fusion is to have an approximate knowledge of the noise model of every sensor. Usually, the biases of every sensor are explicitly computed in the state of the filter. One major limitation of the Kalman filter is that it can only estimate linear systems, both in terms of propagation and measurement. Therefore an extended version of it has been designed for non-linear systems: it linearizes locally the propagation and measurement equations. Many applications and adaptations of the filter, or its extended version, have been presented for inertial sensor fusion [Lefferts et al. 1982] [Julier & Uhlmann 1997] [Marins et al. 2003] [Kiriy & Buehler 2002] [Foxlin 1996].

The **complementary filter** is much more specific to inertial sensor fusion in terms of field of application. The sensors characteristics are taken into account in terms of their frequency performance for orientation estimation. On one hand, gyroscopes are quite accurate but suffer from drift as we need to integrate the measurements to compute orientation. Thus, they perform well to estimate high frequency motions, but poorly for low frequency ones. On the other hand, magnetometers and accelerometers possess heavy noise and perturbations, but their orientation measurements do not drift, as they respectively record the direction of the gravity and magnetic north. Therefore their performance is good in the low frequencies, but poor in the high frequencies. To maximize the potential of both sensors, the complementary filter has two steps: filtering every sensor output according to its frequency quality of estimation, and then combining the filtered signals. For instance, a low-pass filtering is performed on the accelerometer and magnetometer; a high-pass filtering is performed on the gyroscope; then the signals are combined. This type of filtering has been widely applied for inertial sensor fusion [Mahony et al. 2008; Mahony et al. 2005] [Euston et al. 2008] [Fux 2008]. The main advantage of this technique is its low-parameterization, as only cutoff frequencies are to be set. It can also be lighter in terms of computation processing than Kalman filtering.

1.4 Representation of a motion

As the study presented here is focused on motion estimation, a problem that arises is to adopt proper motion model representation. This will impact heavily some algorithmic and implementation choices and limitations. Firstly, the pinhole projective model is introduced, which is the classical model used for embedded video applications. In two dimensions, the problematic revolves around restrictions to the planar motion, going from a pure translation to perspective models, and even some more specific ones. In 3D the main concern is the type of rotation representations that can lead to singularities, difficulties in interpretation, and filtering problems.

1.4.1 Envisioning a 3D world with a 2D sensor

The pinhole camera model is the most applied in the computer vision domain. It describes the mathematical relationship between a 3D object viewed by the camera and its 2D projection on the image, as shown in Fig.1-1. It possesses many limitations: it does not take into account the focus of the camera which creates blur and it does not directly model the discretization that occurs when translating projected image into pixels. In addition, Image distortions due to lenses are not considered. However, this model is considered as a sufficient geometric approximation for many applications [Hartley & Zisserman 2003]. As one can see on fig. 1-1, the real final image plane presents a 180° rotation due to image rays crossing at the pinhole location. To simplify computation, a virtual image plane is often considered in front of the pinhole. All equations presented in this thesis will be based on this virtual image, which will now directly be referred to as the image to clarify and lighten the subject.

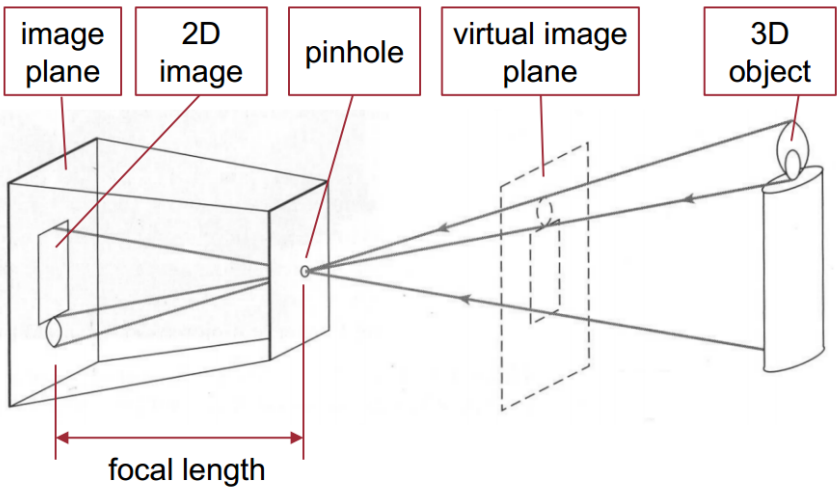


Figure 1-6: A pinhole camera model

We now consider a point X in 3D world homogenous (we add a 1 at the end of the classical coordinates to treat translation using multiplication) coordinates: $X = (x, y, z, 1)^T$. The quantities x, y and z represent the world coordinates of the point. The image projection of the point X is noted I_x with its 2D pixel coordinates $I_x = (u, v, 1)^T$, u and v being the horizontal

and vertical pixel coordinates respectively. A scheme of this representation is displayed on fig. 2-2.

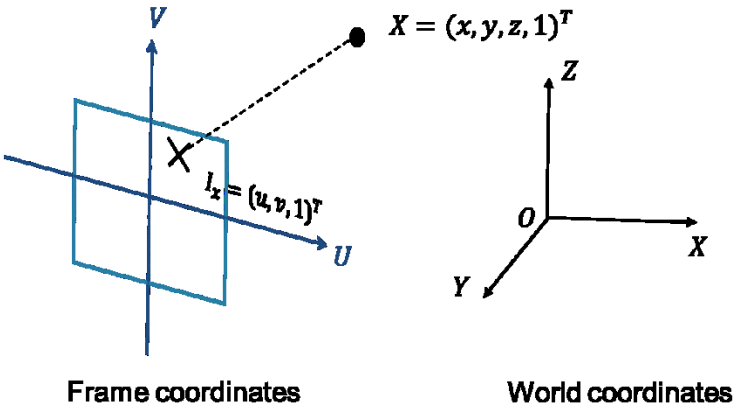


Figure 1-7: Coordinates notation

The pinhole camera model describes the relationship between X and I_x. This is made in two steps. Firstly, we need to explicitly model the geometric position and orientation of the camera with respect to world coordinates. This information is contained in a 3x4 **projection matrix** $\boldsymbol{P} = [\boldsymbol{R}|\boldsymbol{t}]$, where R is a 3x3 rotation matrix that encodes the orientation of the camera, and t a column vector of 3 elements, representing the position of the pinhole center of the camera. Secondly, we need to explicit the transformation of the projection into pixel points. This is modeled by a **camera matrix** $\boldsymbol{K}$. In some studies, K is named the instrinsic matrix.

$$K = \begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad (1)$$

where f is the focal length of the camera, and $(c_x, c_y)^T$ the principal point of the camera, that defines the projection of the camera principal rays into the image plane. Note that a square pixel is considered here, otherwise we would need to define two distinct focal length for the horizontal and vertical axis of the camera. The complete relationship between pixel location I_x and 3D coordinates X is thus:

$$I_x = KPX \quad (2)$$

While one may consider on-the-fly computation of both K and P matrices, in this study we considered that the camera matrix was computed once in a calibration process and then was considered fixed. The method of [Zhang 2000] was applied to every testing device used in the context of the thesis in order to compute the intrinsic camera matrix K.

1.4.2 In-plane motions

A 2D transformation between two frames can be expressed in many different manners. To keep the notation homogenous and simple, we will represent the transformation using the

coordinates' changes of a point. A 2D homogenous point $I_x = (u, v, 1)^T$ in the first frame is mapped to a point $I'_x = (u', v', 1)^T$ in the second frame by the transformation. Most of the following descriptions for motion models is further described in the second chapter of [Hartley & Zisserman 2003].

The first type of motion that can be modeled is a direct translation $T = (T_x, T_y)$. It has a very simple effect on the coordinates:

$$I'_x = \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = \begin{pmatrix} u + T_x \\ v + T_y \\ 1 \end{pmatrix} \tag{3}$$

The main characteristic of this motion model is that it only has 2 degrees of freedom. Therefore it is computable from only one point correspondence from a local motion estimation technique or a global one such as integral projections [Crawford et al. 2004]. The limitation in terms of motion is very restrictive, and makes it only usable for very closely recorded frames. To our knowledge, the translational motion model has mainly been used for video encoding, where every block's motion is estimated with a local translational motion model. This type of model can also be used in panorama and stabilization, if in-plane rotation is not considered.

Another type of 2D motion model is the rotation-preserving isometry, which correspond to an in-plane rotation by an angle θ combined with a translation:

$$I'_x = \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & T_x \\ \sin(\theta) & \cos(\theta) & T_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{4}$$

Only one degree of freedom is added to the translation model, but as a point correspondence provides two pieces of data, two point correspondences are needed to compute the isometry. This motion model is widely used for video stabilization, providing translational and rotational movement estimation that can be filtered. It is also sometimes used in tracking applications, when the size of the object on the image is not expected to change during the sequence.

For non-subsequent image motion estimation, scale changes need to be added to the motion model. This type of model is called a similarity transformation, with a scale change of Z:

$$I'_x = \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = \begin{pmatrix} \mathrm{Z}\cos(\theta) & -Z\sin(\theta) & T_x \\ \mathrm{Z}\sin(\theta) & Z\cos(\theta) & T_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{5}$$

The augmentation of the model with scale opens up many application domains: long term tracking, recognition, etc… See fig. 2-7 for a recognition example based on local matching followed by a similarity motion model estimation. It can be seen that without the addition of the scale parameter, this recognition would have been impossible.

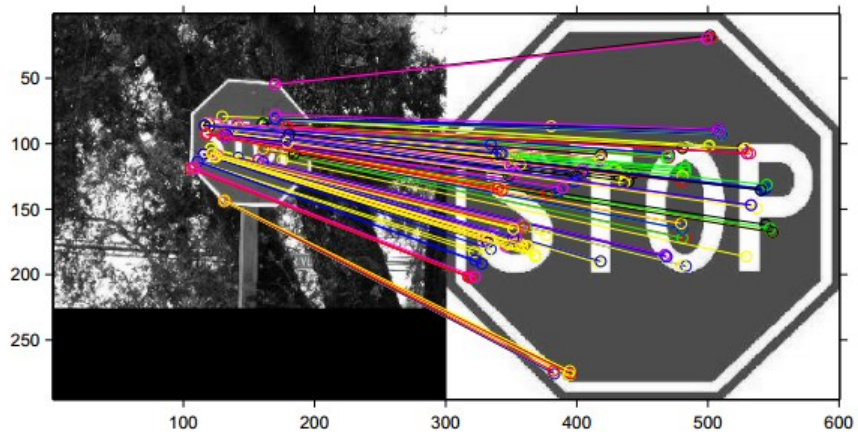


Figure 1-8: An example of similarity motion model applied to recognize a stop sign

Certain types of motions can lead to a deformation in the shape of the image. To include some simple transformations such as stretching or skewing we need to increase the number of parameters in the model:

$$I'_x = \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & T_x \\ a_{21} & a_{22} & T_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \quad (6)$$

This type of representation is an affine transformation. For instance in [Auberger & Alibay 2014], this model is mapped to deduce specific deformations, created by motions recorded with a rolling shutter sensor. The extension to affine model was needed as these distortions do not preserve the shape of the image. As the degree of freedom is increased to 6, three points correspondences are needed to create this type of representation.

The last extension of this type of representation presented here is the projective transformation. The form of the transformation is the following:

$$I'_x = \begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \quad (7)$$

Note than the third coordinate is modified in the final image point I'_x. To retrieve the final location of the point on the image, one should divide the coordinates of the point by w'. This model is needed when modeling "out-of-plane" transformations, for instance 3D rotations. It is useful in applications requiring the tracking of a planar structure moving freely in a scene. More complex 2D motion representations can be used, but their domain of application is more limited in the scope of this study.

1.4.3 Three dimensional representation of motions

3D motion representation is a complex subject. Many types of models exist, but only the most applied in our context of general motion estimation purposes will be displayed here. Rotation

representation will first be described. A discussion is made on how to combine it with translation. We here present the rotation matrix and Euler angles as they will be used in the thesis. Presentation of quaternions and exponential maps are presented in Appendix B:.

1.4.3.1 Rotation matrix

A rotation can be represented as a 3x3 matrix R. The matrix columns are each of unit length and mutually orthogonal, and the determinant of the matrix is +1. This type of matrices constitutes the SO(3) (for special orthogonal) group. Each matrix belonging to this group is a 3D rotation, and any composition of matrices from this group is a rotation. This representation of a rotation is the most direct one to apply, as a 3D point $X = (x, y, z, 1)^T$ is transformed by R to a point $X_{rot} = (x_{rot}, y_{rot}, z_{rot}, 1)^T$ by a simple 4x4 matrix multiplication based on the rotation R:

$$X_{rot} = \begin{pmatrix} R & 0 \\ 0 & 1 \end{pmatrix} X \qquad (8)$$

It must be noted that most of the other rotations representations are converted to a rotation matrix to be applied. The main drawback of the rotation matrix is the complexity of the constraints to keep the matrix in the SO(3) group when applying optimization of filtering techniques. In effect, those techniques will modify the coefficients of the matrix, but it should always be orthonormalized to belong to the SO(3) group. This is done at heavy cost and needs to be performed at each step of the algorithm where the matrix is modified.

1.4.3.2 Euler angles

The Euler angles representation is the most used for 3D rotations. It consists in separating the rotations to a minimal 3 angle values that represent the respective rotations around the axis of the coordinates in a sequential way. They are referred to as the yaw, the pitch and the roll angles. According to the choice of the user, these three values are either expressed in degrees or radians. In order to apply the transformation to a point, the Euler angles are transformed into separate rotation matrices, which are combined to form the total rotation matrix that is then applied to the point. In this study, we will refer to the yaw as α, the pitch as β, and the roll as γ. A big issue in using Euler angles is the necessity to establish a convention on the order of application of the angles. In effect, one can select which angle represents a rotation around an axis, as well as the order chosen to apply them. This can create confusion and misunderstandings. In fig. 2-8, one can see the axes displayed on a smartphone scheme. To specify the convention used in this study: the yaw is the rotation around the red axis, pitch around the green axis, and roll around the blue axis.

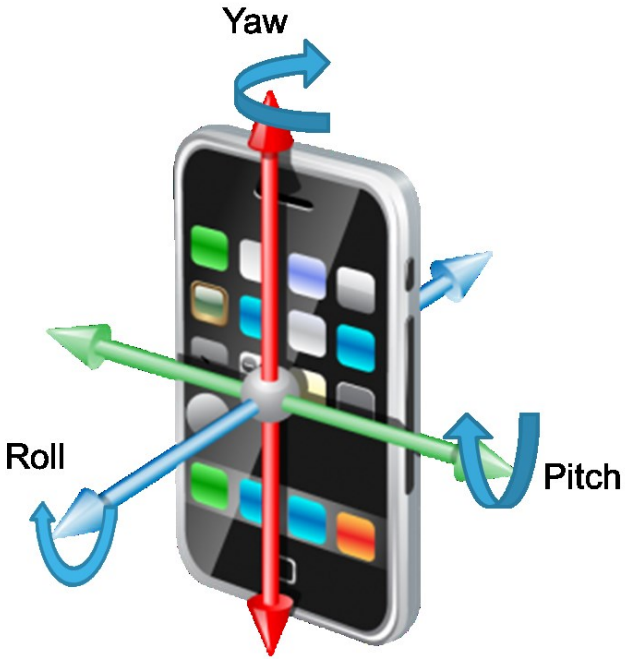


Figure 1-9: Smartphone with axis of superimposed. Image extracted from [4].

Another issue arising from Euler angles application is called the Gimbal lock. It happens when one Euler angle is equal to 90° (this can depend on the convention chosen), which leads to two axes of rotation becoming aligned, as displayed in fig1-10. This degenerate configuration induces a loss of one degree of freedom. Another issue with this representation is that it splits the rotation on the axis, it is difficult to interpolate and differentiate rotations, as it ignores relationship between the several directions.

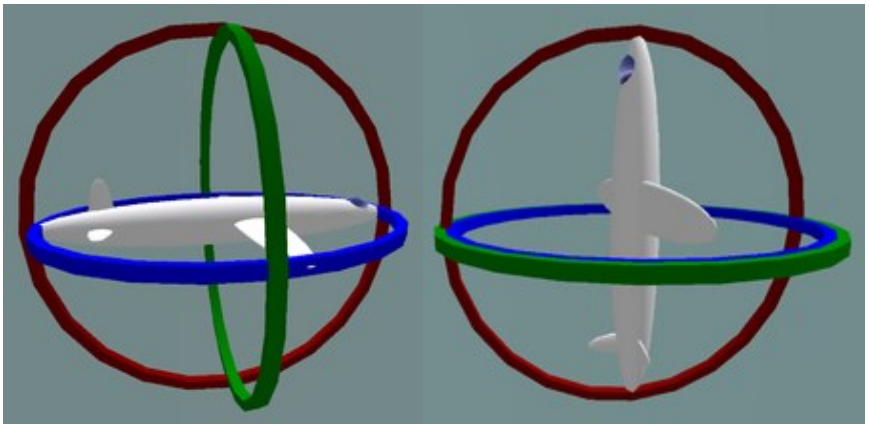


Figure 1-10: Gimbal lock example. On the left, a regular configuration. On the right, one can see that two axes of rotation are aligned, decreasing the number of degrees of freedom. Image extracted from Wikipedia[5]

1.4.3.3 Combination with translation

[4] http://www.legalsearchmarketing.com/mobile-apps/creating-a-mobile-app-for-your/
[5] https://fr.wikipedia.org/wiki/Blocage_de_cardan

A 3D motion is a combination of a rotation and a translation $\tau = [\tau_x, \tau_y, \tau_z]^T$. As seen previously, one must always convert a rotation to a rotation matrix in order to apply it to a point. The complete motion regrouping a rotation R and a translation τ is applied to a point X by:

$$X' = \begin{pmatrix} R & \tau \\ 0 & 1 \end{pmatrix} X \qquad (9)$$

As seen previously, many representations for 3D rotations can be used with various advantages and drawbacks. In SLAM systems, optimization or filtering techniques are applied to retrieve the 3D pose of a device. This often leads to the selection of Quaternion or exponential maps representations that show good properties for differentiation process and do not present disturbing singularities.

1.5 Structure of the thesis

Chapter 2 focuses on motion vector computation. An overview on local methods to compute motion is given, with a focus on feature-based procedures, and especially the ones with the less amount of computation needed. Additions on these techniques using inertial sensors are proposed, and then compared to non-hybrid techniques in terms of performance and computational cost.

In Chapter 3 the problem of computing the camera motion in two dimensions using visual and inertial measurements is studied. An overview of the state of the art techniques is made. Then, a real-time hybrid variation of the RANdom SAmple Consensus (RANSAC) method is proposed. Performance and computational resources are also studied.

Chapter 4 introduces the problem of hybrid localization, or visual-inertial odometry. With an overview on the state of the art techniques presented, hands-on experiments and discussions are made. A novel hybrid localization method is then proposed, which specificity relies on the numerous levels of fusion between visual and inertial measurements, in a very specific manner designed towards embedded platforms. Ground-truth based comparisons are made with state of the art methods, thanks to a setup based on infrared markers and cameras.

Finally, Chapter 5 draws the conclusions and perspectives on the work presented in this thesis.

Chapitre 2: Calcul de point-clés

Below is a French summary of chapter2: Feature estimation.

L'estimation d'une image à l'autre du mouvement de la caméra dans une séquence vidéo est un problème bien connu dans le monde de la vision par ordinateur. Dans une majorité de cas, la première étape de ces techniques est de calculer des vecteurs de mouvements entre deux images. Ceci est réalisé en mettant en correspondances des points d'une image à l'autre.

Pour générer des vecteurs de mouvements, certaines méthodes procèdent à une extraction de points d'intérêts dans l'image, ou points-clés. Ceux-ci sont ensuite stockés sous forme de descripteurs, des représentations haut niveau du point qui sont en général construites afin d'être invariantes à certaines transformations : changement d'illumination, rotation du plan, flou… Nous proposons une amélioration d'un type de descripteur visuel, basée sur les mesures des capteurs inertiels.

Ce chapitre présente tout d'abord les différentes méthodes de génération de vecteurs de mouvements. Les descripteurs peu coûteux en temps de calcul sont étudiés de façon plus approfondie. L'extension inertielle apportée à ce type de descripteur est ensuite présentée. Finalement, une évaluation de ce nouveau système est présentée, avec des considérations sur l'efficacité calculatoire de notre approche.

Chapter 2: Feature estimation

Estimating the frame to frame camera motion in a video sequence is a highly studied problem. It is a key step in many applications: camera stabilization, rolling shutter distortions correction, encoding, tracking, image alignment for High Dynamic Range, denoising… The first step of this type of technique is generally to extract motion vectors between pairs of images. This is performed by putting in correspondences points from one frame to another.

Many factors can impact the performance of these methods. In a sequence, illumination changes can occur, modifying the pixels values from one frame to another. In-plane rotations create another dimension in the problem, which can no longer be solved as a 2D translation estimation. Motion artifacts, such as motion blur or rolling shutter distortions also intervene in the process, creating variation in terms of pixel values and localizations. Finally, scene characteristics can make a great impact on the results of those techniques: a lack of texture in the scene, low-light heavy noise, etc…

An overview of motion vectors generation methods is proposed in section 2.1 . A first category of algorithm makes use of pixel-wise computation. For a set of fixed given points in the first frame, a correspondence is searched in the second frame. This can be performed either for every point, which is generally called optical flow, or in a sparse manner, with techniques such as block matching or tracking. The second category of vector generation techniques consists in extracting points of interest in every frame that are also called **keypoints**, rather than using fixed points (or every point) in the frame. **Descriptors** of each set of points are then computed, which consist in a higher-level, more robust information on the surrounding of the keypoint. Correspondences are then drawn between these two sets of points, in an operation known as matching.

In section 2.2 , the focus is on a deeper look into light keypoint detection and description techniques. In the context of embedded platforms, it is suitable to spend as little resources as possible on the computation of motion vectors. Therefore, the focus of this thesis for feature estimation was centered on as light as possible methods.

A novel technique of light keypoint description extension with inertial sensors is described in section 2.3 . The main contribution in this scope is to study the possible impact of inertial sensors for description techniques, and to provide a valuable addition to binary descriptors. Robustness to geometric variations such as in-plane rotation or viewpoint changes is improved.

With the global scheme of the method described, section 2.4 presents implementation and results of the process. Results show the improvement in robustness induced by the techniques. Implementation approximation is proposed to save as much computation as possible.

Finally, conclusions are drawn in section 2.5 .

2.1 From images to motion vectors

Two main methods exist to compute the motion vectors:

- Computing the position of a particular set of points from one frame to the next. It can be done in a dense manner (every point position is computed) or sparsely.
- Extracting points of interest in both frames, and intending to put them in correspondences, which is also called matching.

After performing the motion vectors computation, global camera motion is estimated with robust techniques. More recent techniques intend to combine the inertial measurements with the motion vectors to provide a robust and accurate estimation of the motion.

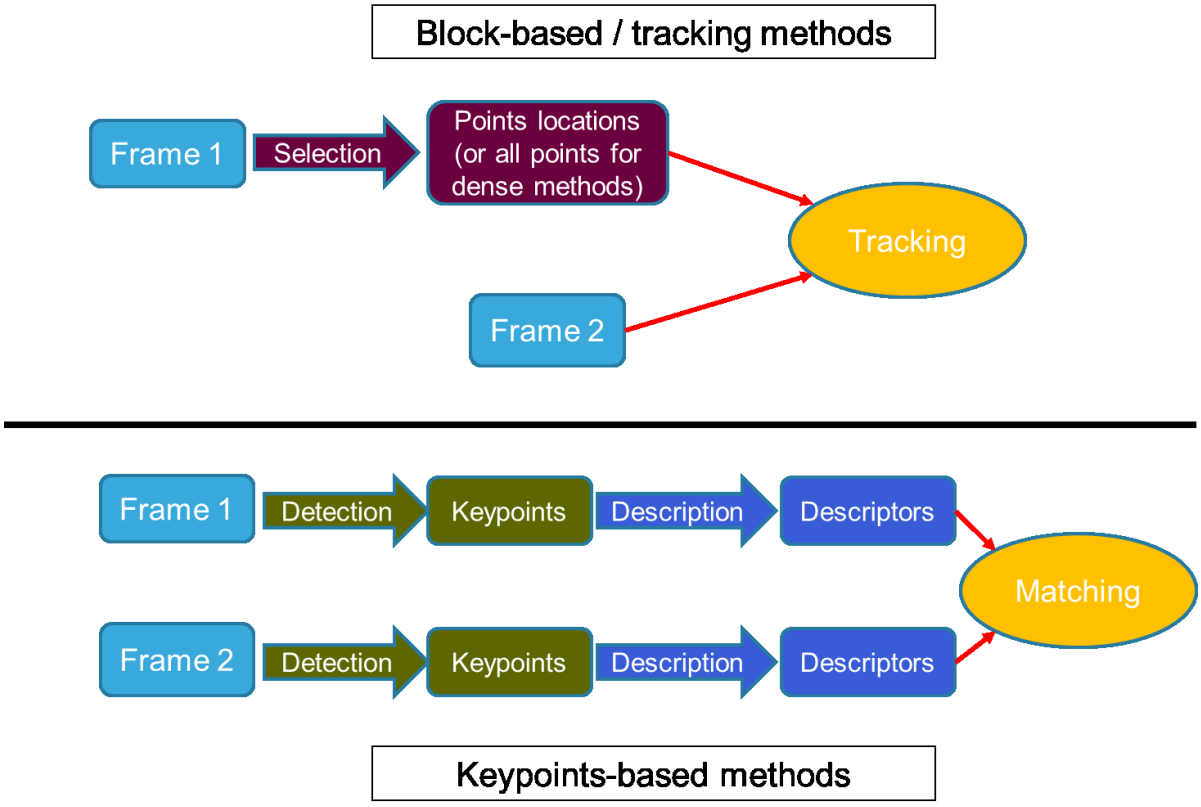


Figure 2-1: The two main types of motion vector computation techniques.

In Figure 2-1, one can see the two types of motion vector creation techniques. For block-based / tracking techniques, the first stage is selection of points: either considering points at a fixed location on the frame (with a grid pattern for instance), detecting keypoints, or every points for dense methods. Then the technique consists in finding the corresponding locations of the selected points in the second frame. For keypoints-based methods, keypoints are detected on every frame. Description algorithms are performed on the keypoints locations, creating descriptors, which are matched from one frame to another to generate motion vectors.

2.1.1 Block-based and global methods

While some techniques intend to find motion directly based on the whole image [Crawford et al. 2004], most of the recent techniques compute local motion vectors as a first step of a motion estimation procedure. A motion vector describes a mapping of a point in the previous frame to the current one. Block matching methods compute a difference function between two pixels based on their surroundings, which is a square area around each pixel called blocks. The main consideration in this type of algorithms is the number of candidates to test, as well as their locations. Differential methods intend to compute the motion for every pixel of the frame, by minimizing an energy that incorporates local and global terms [Horn & Schunck 1981].

2.1.1.1 Block matching techniques

Block matching algorithms computes the similarity of two pixel blocks based on a difference function, either the Sum of Square Differences (SSD) or Sum of Absolute Differences (SAD). Then the motion vector computed is considered as an estimation of the motion of the center of the block. For a pixel point $p = [u, v]^T$ in the previous frame, we note its intensity $I(u, v)$. The difference of this point p with another point in the current frame $p' = [u', v']^T$ is computed as $D(p, p')$:

$$D(p,p') = \sum_{i=-N}^{i=N} \sum_{j=-N}^{j=N} diff(I(u+i, v+j) - I(u'+i, v'+j)) \quad (10)$$

Where the function $diff$ is either the absolute or square difference. This function D is computed on a certain amount of locations, and the found minimum of the function is selected. The motion vector $mv(p) = [u' - u, v' - v]$ is therefore created.

The SSD (or SAD) score presented can be used as a starting point in order to compute similarity between two points. However, it is not robust to illumination changes, neither to a shift nor to affine transformation. That is why a zero-normalized SSD score is preferable:

$$ZNSSD(p,p') = \frac{1}{(2N+1)^2} \sum_{i=-N}^{i=N} \sum_{j=-N}^{j=N} \left(\frac{I(u+i, v+j) - m_p}{\sigma_p} - \frac{I(u'+i, v'+j) - m_{p'}}{\sigma_{p'}}\right)^2 \quad (11)$$

Where m_p and $m_{p'}$ are the mean values of pixel blocks around points p and p' respectively, and σ_p, $\sigma_{p'}$ the standard deviation of those patches. While this technique is effective in order to compare points, it is not robust to rotations, scale, or viewpoints changes.

The key step of this method is to carefully choose the tested points in the previous frame. Many types of techniques exist to choose points, from spatial ones [Koga et al. 1981][Li et al.

1994][Po & Ma 1996][Zhu & Ma 2000], to spatial-temporal strategies [de Haan et al. 1993]. These methods were mainly developed towards video encoding.

2.1.1.2 Optical flow differential methods

Optical flow differential techniques intent to minimize the brightness constant constraint of the neighborhood around the pixels with derivate-based methods. Rather than only testing a few candidates, the **Lucas Kanade Tracker (LKT)** [Lucas & Kanade 1981] performs a gradient descent on the neighborhood around the point, inferring that the flow do not vary much in this location. A major improvement was proposed in [Bouguet 2001], that implements a pyramidal version of the tracker. It computes the gradient and optimizes the position on a reduced size image. This computed position is then taken as the starting point for the next image pyramid level, going from a very small image up to the real image size. Due to heavy cost of computation, adaptations have been made to port the algorithms on a graphical processor [Kim et al. 2009].

On another hand, the pioneer work of [Horn & Schunck 1981] displays an iterative method to compute the optical flow for every pixel, minimizing an energy term for each motion vector in both horizontal and vertical directions. This term relies on a combination of the brightness constancy constraint and the smoothness expected in the optical flow. A weighting value balances the importance between the smoothness and brightness constancy constraints. In a similar fashion to the LKT, the heavy computation cost has led to processing time reduction techniques [Bruhn et al. 2005].

The main difference between the two presented types of methods is that LKT strategy only relies on the neighborhood of a pixel, while Horn & Schunck variational methods add global constraints to the optical flow fields. In the literature, LKT is used when tracking a sparse set of points. When needing a dense field, variational methods are preferred.

2.1.2 Keypoints

The main drawback of previously presented methods for pixel motion estimation, is that every pixel does not carry the same amount of useful information. For instance, estimating the motion of a white point on a white wall is much more challenging than a black point on the same wall. If a motion needs to be estimated from two images that present changes in terms of conditions and location, we need robustness to various transformations in our estimation approach: illumination changes, rotation, scale… Approaches of feature extraction have been designed with the goal of finding locations in the images that carry most information and distinctiveness. Many types of features exist, including points, blobs, edges… In this study we restricted our interest to points and blobs that are present in most types of sequences which makes them suitable for embedded applications. These points of interest are called keypoints.

An overview of feature detection strategies is presented, followed by feature description techniques. Then a short discussion is made about matching vs. tracking strategies. Finally, an overview of existing hybrid methods for keypoints computation is done.

2.1.2.1 Detection: From corners to Local extrema detection

One of the first works done in keypoint detection has been demonstrated in [Moravec 1980] which served as base for the Harris corner detector [Harris & Stephens 1988]. It is based on an auto-correlation function a of a point $p = [u, v]^T$ and a shift $[\Delta u, \Delta v]^T$:

$$a(p, \Delta u, \Delta v) = \sum_{i=-N}^{i=N} \sum_{j=-N}^{j=N} (I(u+i, v+j) - I(u+\Delta u+i, v+\Delta v+j))^2 \qquad (12)$$

If this auto-correlation is small in every direction, this translates a uniform region with little interest. Only a strong value in one direction most likely indicates a contour. If every direction displays strong values however, the point is considered a keypoint. With a first-order Taylor approximate, the auto-correlation matrix can be expressed in function of spatial derivate of the image. The keypoint evaluation is then made with regard to the eigenvalues λ_1, λ_2 of that matrix. The corner-ness function is:

$$f(p) = det(a(p)) - k\big(trace(a(p))\big)^2 = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \quad (13)$$

If this value at pixel p is higher than a threshold and higher than cornerness function f evaluated on neighborhood points, the point is considered a corner. The threshold can be set in function of the total desired number $N_{corners}$ of corners, or an absolute quality desired. The fact that the detectors consider all directions of a gradient matrix induces its robustness to illumination changes as well as in-plane rotations. Other methods have been designed based on the gradient matrix to detect corners [Förstner 1994], some extending the Harris detector to make it scale invariant [Mikolajczyk & Schmid 2001]. [Rosten & Drummond 2006; Rosten et al. 2010] presented Features from Accelerated Segment Test (FAST), a very light extractor in terms of computational time that will be further described in section 2.2.1 . The FAST keypoint extractor is based on a number of continuous pixels on a circle with a marked difference with the center.

Several detectors have been developed to cope with scale invariance. They consist in searching the scale-space dimensions of the image and finding the extremas of an operator, which can be the gradient, Laplacian, etc… The image is first convolved by a Gaussian Kernel to smooth noise. Values are then normalized with respect to scale, and a point that possesses the highest absolute value of the neighborhood is then considered an interest **blob** (a keypoint with higher scale than just a pixel). The Laplacian has been demonstrated to be the best operator to choose [Lindeberg 1998].

In this context, the SIFT detector is presented in the work of [Lowe 2004], making use of the difference of Gaussians approach between scales. On Figure 2-2, an example can be seen of a feature detected in a difference of Gaussians image. For every scale, the Laplacian operator is approximated by a difference between two Gaussians smoothed images with different values for Gaussian smoothing. Maxima / minima are then detected by comparing every point with its direct neighborhood in the space scale dimensions, reducing the number of comparisons needed to only 26 neighbors.

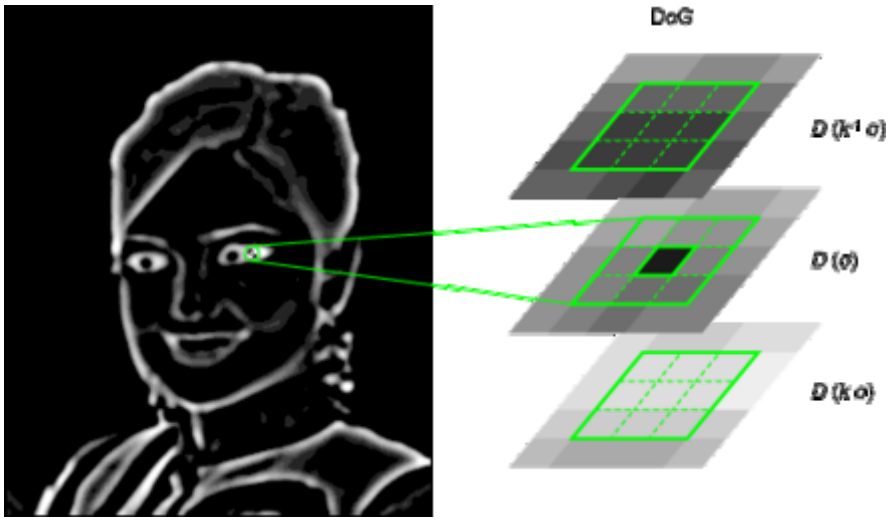


Figure 2-2: Scale space detection based on difference of Gaussian, extracted from [6].

To speed-up processing, [Bay, Tuytelaars 2006] presented SURF, a method based on the Hessian matrix of a point p at scale σ:

$$H(p,\sigma) = \begin{pmatrix} I_{xx}(p,\sigma) & I_{xy}(p,\sigma) \\ I_{xy}(p,\sigma) & I_{yy}(p,\sigma) \end{pmatrix} \quad (14)$$

Where $I_{xx}(p,\sigma)$, $I_{xy}(p,\sigma), I_{yy}(p,\sigma)$ are the second-order derivate of the image in their respective directions. The value used to perform blob detection is the determinant of the Hessian matrix. A non-maxima suppression by comparison of the neighborhood in the space scale dimensions is then performed, in a similar fashion to the SIFT method. SURF was mainly developed to propose a less expensive alternative to SIFT, using approximations to perform similar treatments. Mixed techniques also exist, using different operators to compute the scale and space selections [Mikolajczyk 2004].

To increase robustness to viewpoint changes, affine-invariant detectors were developed, using second moment matrix to estimate affine neighborhood [Lindeberg 1998], before applying it to the scale invariant blobs [Mikolajczyk & Schmid 2002]. Other techniques apply a connected components strategy in thresholded images in order to detect blobs [Matas et al. 2004]. While showing excellent results in terms of repeatability and robustness to viewpoint changes, these detectors have been measured as prohibitively expensive for real-time

[6] https://fr.wikipedia.org/wiki/Scale-invariant_feature_transform

applications in multiple comparisons [Mikolajczyk et al. 2005] [Moreels & Perona 2007] [Gauglitz et al. 2011].

2.1.2.2 Feature description

In the same efforts to improve detection robustness, keypoints descriptors techniques were developed to provide invariant representation of the surroundings of the desired area. Early techniques have been presented in order to deal with this issue [Freeman & Adelson 1991] [Schmid & Mohr 1997], but the SIFT descriptor [Lowe 2004] was the first widely used robust keypoint descriptor. The descriptor is based on the scale extracted from the keypoint (see previous section). It determines a dominant rotation in the scaled patch around the point. Image gradients are then computed in zones, and a soft assignment is made to map them into 8 discrete orientations (see Figure 2-3). The normalized histogram of gradients strength into each direction for every spatial zone constitutes the descriptor. It is a 128-dimensional vector of continuous numbers: 4x4 spatial zones times 8 orientations. To compute the similarity between two SIFT descriptors, one simply has to compute the L2-norm between them.

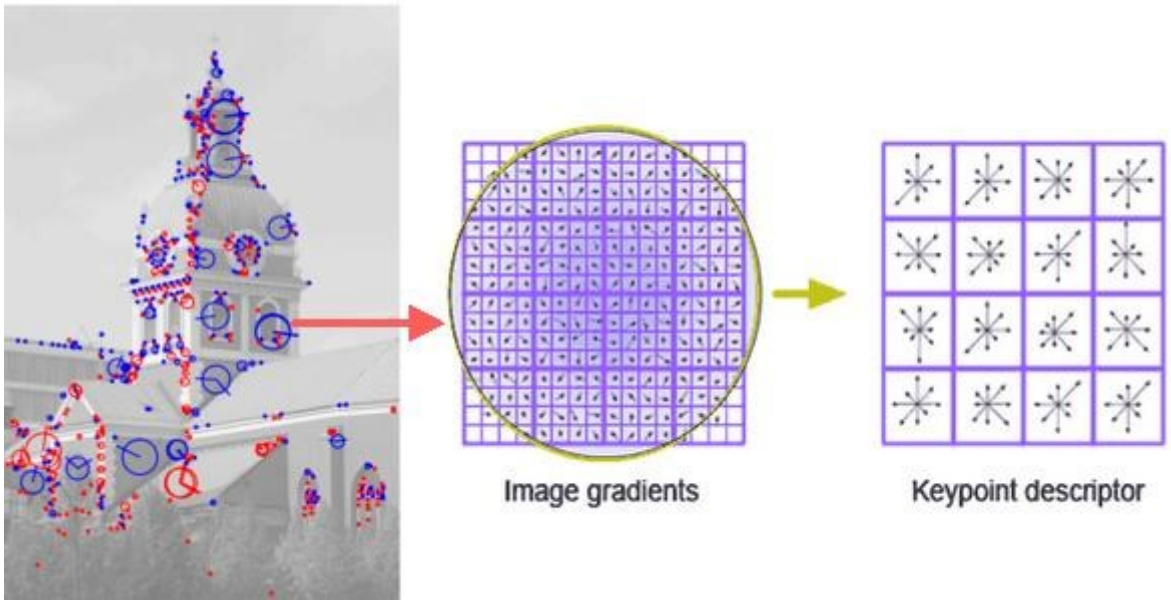


Figure 2-3: Feature description with SIFT, courtesy of [7].

SIFT is still a widely used descriptor, its main drawback being that the L2-norm between two 128-dimensional vectors can become very expensive to compute, especially when computing similarities between many points. [Ke & Sukthankar 2004] and [Mikolajczyk & Schmid 2005] introduced Principal Component Analysis (PCA) methods to reduce the dimension of the vector, decreasing the computation of matching. In the same spirit than extractors, [Bay, Tuytelaars 2006] presented approximations by box filtering of the description with SURF.

Taking into account the computer quickness in computing the binary Hamming distance (just a XOR operation followed by a bit count), recent techniques were developed to produce binary

[7] http://www.codeproject.com/Articles/619039/Bag-of-Features-Descriptor-on-SIFT-Features-with-O

descriptors. Binary Robust Independent Element Features (BRIEF) [Calonder et al. 2010] utilizes pairwise comparisons along a determined pattern to build the binary string descriptor. The pattern is built with a random generation technique based on a Gaussian sampling that has been proven superior against other sampling strategies [Calonder et al. 2010]. This descriptor will be further described in section 2.2.2 . As BRIEF is not robust to scale or rotation changes, it is mostly used for video sequences needing frame to frame matching. To circumvent this, propositions have been made to bring invariance to rotation and scale to the detector with ORB [Rublee et al. 2011] and BRISK [Leutenegger et al. 2011]. Other methods used a regular pattern to perform the comparisons either for dense matching purposes [Tola et al. 2010], to avoid sampling effect [Leutenegger et al. 2011] or to approach the computation to human retina interpretation [Alahi et al. 2012].

The following table summarizes the invariances of some descriptors presented:

Technique	In-plane rotation	Scale invariant	Perspective invariance	Computational cost	Matching norm used
SIFT	++	++	+	--	L2
SURF	++	++	+	-	L2
BRIEF	-	--	-	++	Hamming
ORB	++	--	-	+	Hamming
BRISK	++	++	-	+	Hamming

Table 1: Summary of the performance of some keypoint descriptors. ++ indicates strong performance in the considered category, down to – for inefficiency.

2.1.2.3 Methods using hybrid keypoints

To further improve robustness to rotation of the descriptors, [Kurz & Ben Himane 2011] proposed a gravity oriented version of SIFT. Rather than performing a visual technique to extract the dominant orientation of the patch, the readings of an accelerometer are used. This allows further discrepancy of the patch to transformations, as well as slightly reducing the computational cost. In Figure 2-4, one can see the benefit of orientating the descriptor with respect to gravity on a synthetic example. On the left, one can see that with visual information only, every window's corner would have been described similarly along the dominant gradient. With the addition of gravity direction on the right, their descriptors are effectively modified, allowing to distinguish one from another. In further work [Kurz & Benhimane 2011; Kurz & Benhimane 2012] the utility of gravity oriented features has been demonstrated for more specific augmented reality applications.

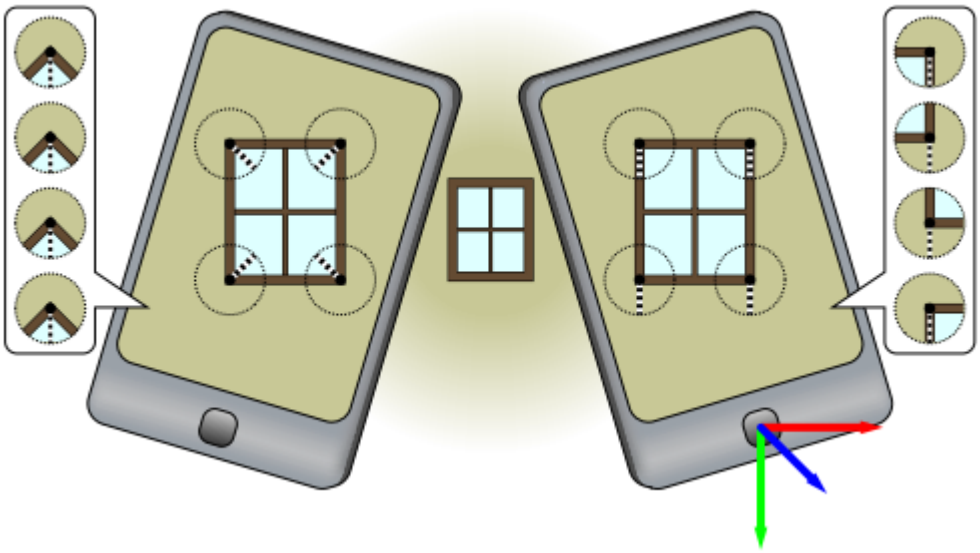


Figure 2-4: Gravity benefit on keypoint discrepancy, extracted from [Kurz & Ben Himane 2011]

[Hwangbo et al. 2009] directly apply the gyroscopes readings in order to change the window of research of the LKT algorithm. With the yaw and pitch measurements they pre-set a translation value to the research window of every point, while the roll is applied to rotate it. This makes the LKT much more robust to quick motions.

2.1.2.4 Keypoint state of the art: discussion

As seen in the various techniques developed in the literature, inertial sensors bring an information on the orientation of the smartphone that is used to increase robustness to geometric transformations. They can also be used to predict the motion of the platform to adapt the search window of a technique. However, they do not bring any valuable addition to scale computation of a keypoint, which can only be computed from visual sensor recordings.

As this study targets real-time performance for applications with high frame rate (up to 60 fps on modern smartphones) and resolution, an emphasis on low-cost technique selection has been stated. The addition of inertial sensors has been proven efficient for classical descriptors such as SIFT [Kurz & Ben Himane 2011], or heavy tracking strategies such as the LKT [Hwangbo et al. 2009].

To provide maximum efficiency of computation, we decided to utilize binary descriptors and light detectors (such as FAST) for the applications developed in this thesis. An example of processing time comparison between classical descriptors such as Upright SURF (U-SURF), which is a non-rotational invariant variation of SURF, and BRIEF is provided in [Calonder et al. 2012]:

Technique	BRIEF (32 Bytes version)	U-SURF (64 values version)
Description time (ms)	6.81	101
Matching time (ms)	0.833	28.3

Table 2: A comparison of U-SURF and BRIEF computational cost on a CPU.

The measurement of processing time were recorded on a CPU. This demonstrates how the binary descriptors are much lighter to compute than classical ones, by a few orders of magnitude. This allows to integrate more points at the same cost, compared to more complex techniques. However, binary descriptors and light detectors also raise some drawbacks:

- Descriptors are binary and not very flexible nor interpretable: one cannot extract information or modify the descriptor directly.
- As matching techniques will be used, the error on measurement is not easy to model, which can be a problem for filtering or optimization techniques that may infer a certain behavior of the error in order to perform properly.
- Some of the binary descriptors are known for presenting slightly lower robustness to geometric changes than more complex ones.

In the next sections, detailed processing of light extraction and description methods will be displayed. Then, two examples of possible improvements with inertial sensors addition developed in this thesis will be presented, with several tests to check the efficiency of the modifications.

2.2 Light detectors and binary descriptors

In the recent years, strategies aiming at detecting and describing keypoints with minimal computational cost were developed. The FAST detector and the binary descriptors are the most applied, as they provide simple yet effective methods to perform local motion estimation.

2.2.1 FAST feature detector

[Rosten & Drummond 2006] introduced FAST, an algorithm to extract keypoints designed to be the lightest possible strategy, sparing computational resources for other tasks. It is a corner detector based on the Bresenham circle of radius 3 around a point p, which is simply the "pixelized" version of a circle. This circle contains 16 points, if a certain amount N_{radius} of contiguous points' intensities are all either lower or brighter than the point p (plus / minus a threshold τ), the point p is considered a corner. An example of a detected corner is displayed on Figure 2-5.

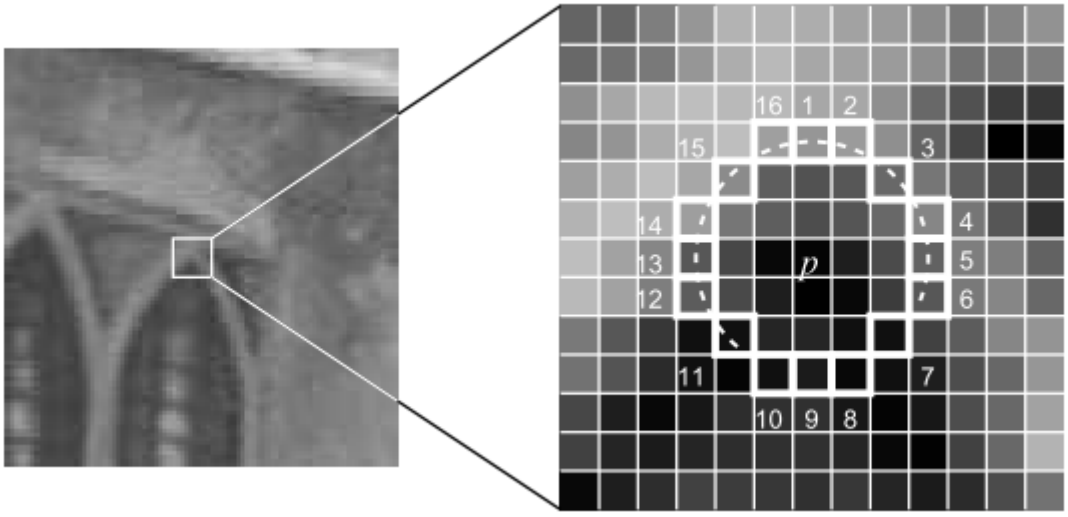


Figure 2-5: A keypoint detected as a corner, image extracted from [Rosten & Drummond 2006]

A rigorous analysis of keypoints repeatability and processing time has been performed in [Rosten et al. 2010], with an addition of a machine learning based version of the FAST detector. It has been shown that using the $N_{radius} = 9$ (9 contiguous points are used to detect a corner) was a very good choice compared to other values, such as $N_{radius} = 12$ [Rosten & Drummond 2006]. A selection of specialized decision trees to modify the execution of tests and masks in FAST has been demonstrated in [Mair et al. 2010], which decreases the average amount of computational time per point extracted. While the modification of test executions is often utilized in further work [Leutenegger et al. 2011], the 9-16 configuration is retained.

The main drawbacks of this feature detector are:

- Reliance on a threshold τ that highly influences the performance of the extractor.
- No detection on multiple scale, it is only a corner detector. It is however possible to achieve multiple scale detection thanks to pyramidal strategies like in BRISK [Leutenegger et al. 2011].
- It is highly sensitive to image noise, as a single pixel that changes value can break the contiguous pixel arc.

2.2.2 Binary keypoint descriptors

The BRIEF descriptor technique has been introduced in [Calonder et al. 2010]. It is based on a pattern of point correspondences around the point p that we want to describe. Many types of patterns exist, see Figure 2-6 for examples of the several patterns tested in BRIEF.

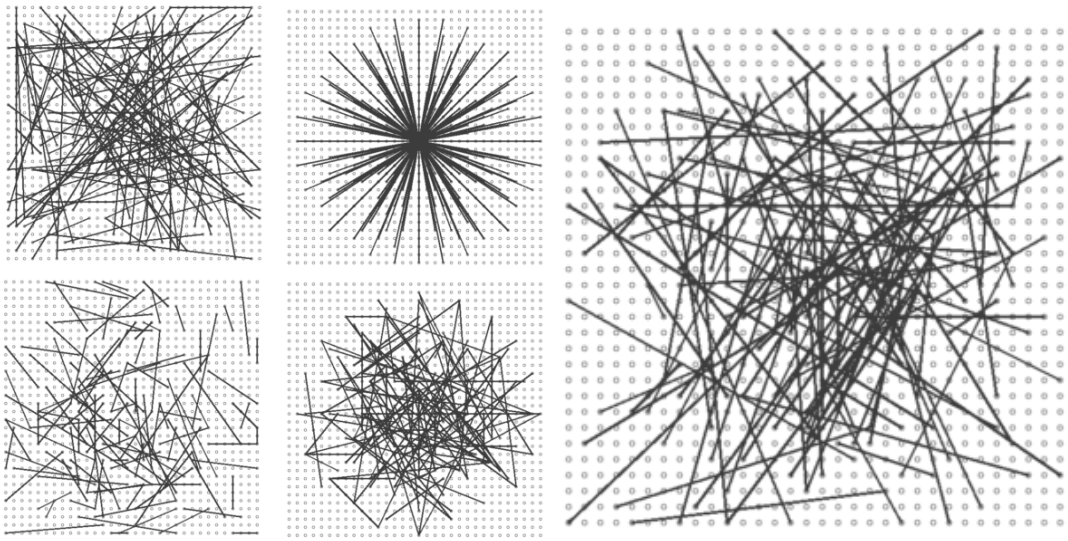


Figure 2-6: Several patterns possible for BRIEF. Each line represent a couple of points, which comparison will lead to a binary element in the descriptor. On the left, several patterns possible, on the right, the selected one that has proven to be more efficient

Given a pattern of predefined pairs (C_i, C_i') in the pixel patch around the point p, the BRIEF descriptor D_p is built as follows:

1) Every pixel in the patch is smoothed with a 9x9 Gaussian Kernel with a σ value of 2 (multiple values of the parameters where tested, these ones had the best results [Calonder et al. 2010])
2) For every pair (C_i, C_i'):
   if $I(C_i) < I(C_i')$ then $D_p(i) = 1$, otherwise $D_p(i) = 0$

where $D_p(i)$ is the value of the binary descriptor D_p at index i. The total number of pairs n_d is very often a multiple of 8, therefore the descriptor has a length of $n_d/8$ bytes. The common values for n_d are 128, 256 or 512. Comparisons between two descriptors are made with the Hamming distance, which is simply computed with a XOR operation followed by a bit count that can be implemented very efficiently on a classical processor.

The locations of the points pairs where determined with a Gaussian random pattern, that has been demonstrated as the most efficient (see fig.2-6). As a Gaussian blur is applied before performing the comparisons, the BRIEF descriptor is robust to noise. However, it is not robust to in-plane rotations. With its high speed of computation and matching, it is a very adapted descriptor for video sequences frame-to-frame local motion estimation, but is not the ideal descriptor for longer term estimation.

Two techniques have been developed in order to bring more invariances to the BRIEF technique: ORB [Rublee et al. 2011] and BRISK [Leutenegger et al. 2011]. ORB was designed to improve the FAST / BRIEF local motion estimation technique to cope with in-plane

rotations, thanks to a dominant orientation estimation, that is used to rotate the BRIEF pattern according to it. To save computational time and avoid to recomputed a pattern for every point, a few discrete angles values are applied and computed, and are used to perform the rotated BRIEF descriptor technique. BRISK makes use of a scale space search with the FAST extractor, orientation is detected with similar technique as ORB. Then a different pattern from BRIEF is applied, every point is blurred with a different Gaussian Kernel according to given locations: see Figure 2-7, red dotted circle shows the kernel size for every point on the pattern, which are the blue points.

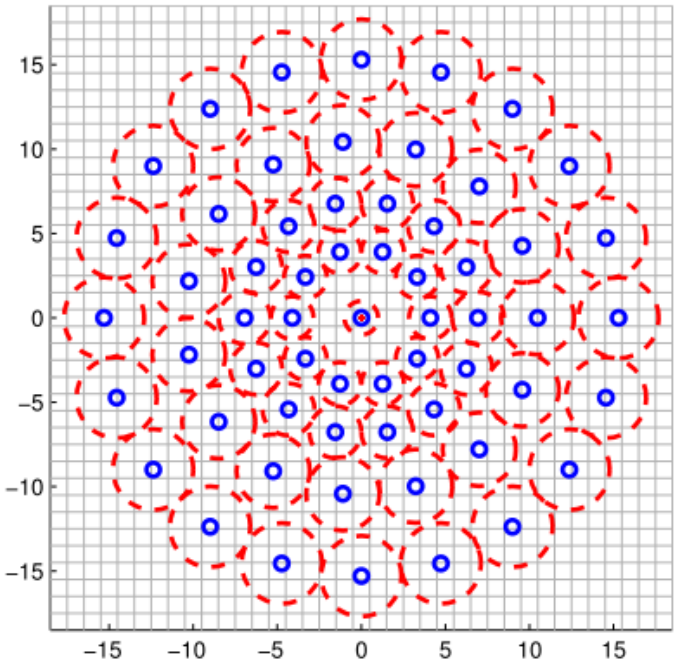


Figure 2-7: The BRISK pattern. Each blue circle represent a point, the red circle is the used kernel to blur it [Leutenegger et al. 2011].

2.2.3 Blur experiments on BRIEF

Now that an overview of the literature and in-depth description of the FAST and BRIEF techniques have been performed, improvements using inertial sensors on binary feature descriptors in this thesis will be presented.

After tests and manipulations on BRIEF used on recorded sequences, a first issue identified was its relative lack of robustness to motion blur. Motion blur is a well-known issue in the computer vision domain. If a motion occurs during the exposure time of the camera, a pixel will capture a combination of the light that was supposed to hit nearby pixels, according to the motion that occurred. An example of this can be seen on Figure 2-8. On the left, we can see three pixels recorded, with a fixed camera and non-moving objects. On the right, due to motion during the capture, the light is distributed on many pixels.

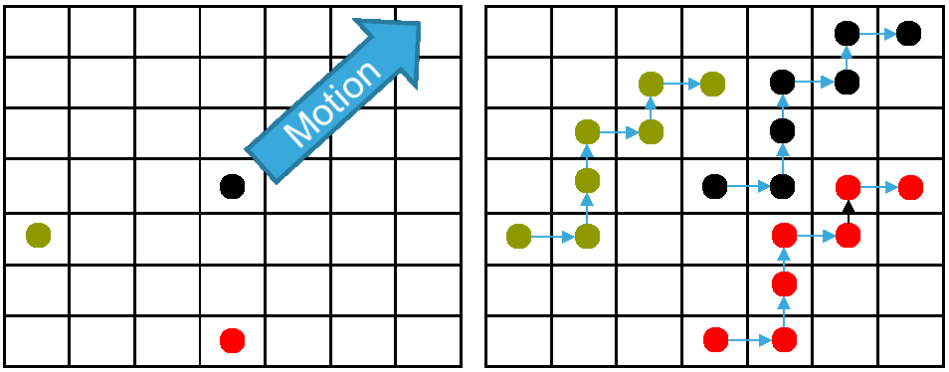


Figure 2-8: the difference between a classical recording and the effect of motion on it.

The Point Spread Function (PSF) describes the motion that occurred during the capture. It is necessary to have an estimation of this function in order to correct blur and restore sharpness [Bovik 2010] . In the context of keypoint description, an idea that we wanted to implement was to adapt the blur applied in the BRIEF descriptor with respect to this PSF function.

The first step is therefore to have an estimation of the PSF. To perform this, we use the gyroscope readings that were measured during the capture of two frames. We convert the yaw and pitch speed of rotations into motion vectors using the factors ρ_x, ρ_y proposed by [You, U. Neumann, et al. 1999]:

$$\rho_x = L_x/2\, tan^{-1}(L_x/2f_x) \qquad (15)$$

where L_x is the resolution of the image in the horizontal direction, and f_x the focal length of the camera in the horizontal direction. ρ_y is computed in a similar fashion with values for the vertical direction. Then, a yaw variation α and a pitch variation β can be converted into a visual motion vector $T = (T_x, T_y)^T$:

$$T_x = \rho_x * \alpha \qquad (16)$$
$$T_y = \rho_y * \beta \qquad (17)$$

Given n_r inertial readings between the recordings of two frames, we build an estimation of the PSF by placing the motion vectors T_i measured by the gyroscope on a pixel grid, and interpolating the positions to generate a PSF. Each pixel on the way of the motion vector T_i is marked with a value τ_i, that is equal to $1/(n_r v_i)$, where v_i is the number of marked pixels for each vector T_i. An example for two motion vectors recorded is displayed on Figure 2-9:

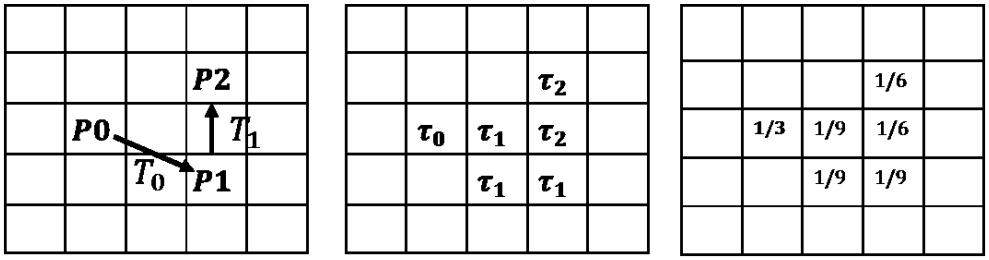


Figure 2-9: An example of motion vectors estimated and associated PSF.

Now that an inertial estimation of the PSF is available, we want to de-convolute the Gaussian kernel utilized in the BRIEF technique with the estimated PSF. It can be done by computing the Fourier's transforms of the Gaussian Kernel and PSF. Then we simply multiply the transform of the Gaussian kernel by the inverse of the PSF, and transform back this combination. Using this technique, we can compensate the motion blur in the first step of the BRIEF algorithm, making it more robust to motion blur.

However, this idea did not work well in practice, even deteriorating matching results. In effect, the computing Fourier's transforms is very expensive. Furthermore, the MEMS gyroscope did not present enough accuracy (~5-10 pixels) when estimating the PSF, which led to wrong changes in the blurring stage combination. To conclude, changes can be applied to the BRIEF technique to make it more robust to motion blur using inertial sensors in a smartphone, but it would need very accurate gyroscopes to bring improvements.

2.3 Improving BRIEF descriptors robustness to rotations with inertial sensors

As presented in section 2.2.2, the BRIEF algorithm does not cope well with rotations. It is neither robust to in-plane rotation nor viewpoint changes (yaw and pitch variations). While this lack of robustness is not necessary for a frame-to-frame memory-less motion estimation, it is problematic for longer terms tracking of points. If an update of the descriptor is needed at every matching step, a simple mismatch can lead to wrong data association. In this context, an improvement upon the original BRIEF descriptor has been designed in order to increase its robustness to geometric changes.

2.3.1 Design

To improve the geometric invariance of the BRIEF with the addition of inertial sensors, we need to define clearly the needs and constraints for our approach. For embedded applications, where resources of computation are limited, we want an approach that does not add too much complexity to the BRIEF descriptor. The main context of application for a more robust BRIEF version with respect to geometric changes is long-term keypoint matching. This could find applications in techniques such as tracking, SLAM, augmented reality, object recognition, etc…

2.3.1.1 Global scheme of the technique

In every rotation invariant descriptor, common steps are performed: find a dominant orientation of the pixel patch and rotate it along this axis, then use a description method. The main downside of this strategy is the cost it generates: one must add a dominant orientation computation step and rotate every patch. In ORB [Rublee et al. 2011], the BRIEF descriptor is sampled around a few orientations to avoid performing the rotation step for every point. While an efficient way to deal with computation load, this generates inaccuracies when the orientation is not very close to the interpolated values.

Inertial sensors offer us a direct orientation estimation in 3 dimensions. This implies that thanks to inertial measurements, one is able to compute an absolute orientation of the descriptor. To perform invariance with regard to rotation, a first idea would be to rotate the image with the orientation information, and then use a detector / descriptor strategy. However, we believe that it is not a very efficient way to use this information. Furthermore, with high yaw / pitch angles, the image would become too flat to bring valuable visual information.

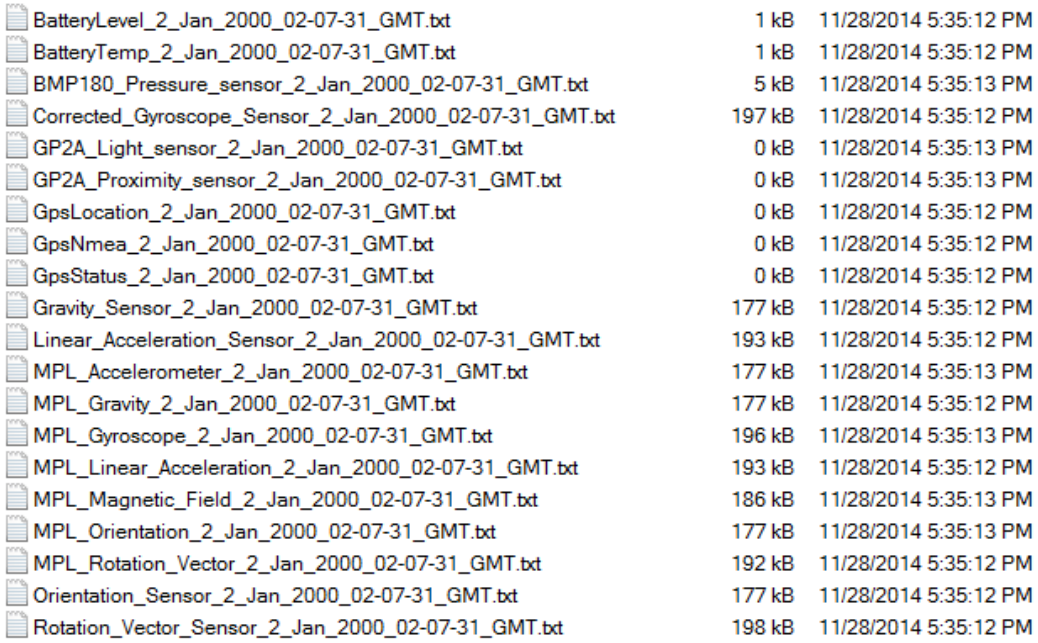
One characteristic of binary descriptors is to use a fixed pattern of point's pairs to apply comparisons. Our idea is to directly use the 3D rotation information on the pattern of point's pairs. This would allow much less computations than a full frame rotation, as the BRIEF pattern can contain up to 512 points, while for instance a 1280x720 pixels frame presents 921600 points! The global scheme of the techniques would be:

1) Record frame and inertial measurements
2) Apply inertial orientation estimation to rotate the binary pattern
3) Perform FAST detection on the image
4) Use the rotated binary pattern to compute new descriptors
5) Match the new descriptors with the descriptors from previous image

2.3.1.2 Inertial sensors use

In modern smartphones or tablets, inertial sensors such as gyroscope, magnetometer and accelerometer are present and integrated, with inertial fusion algorithms often already implemented in hardware that computes orientation estimation. In our experiments, we used a Samsung Galaxy ©, with a software that records video stream along with all sensors data available: the Cellbot data logger[8]. This software shows us all hardware sensors that we are directly able to record. In Figure 2-10 one can see an example of all available sensors on the platform.

[8] http://code.google.com/p/cellbots/downloads/detail?name=CellbotsDataLogger_v0.9.7_lite.apk



BatteryLevel_2_Jan_2000_02-07-31_GMT.txt	1 kB	11/28/2014 5:35:12 PM
BatteryTemp_2_Jan_2000_02-07-31_GMT.txt	1 kB	11/28/2014 5:35:12 PM
BMP180_Pressure_sensor_2_Jan_2000_02-07-31_GMT.txt	5 kB	11/28/2014 5:35:13 PM
Corrected_Gyroscope_Sensor_2_Jan_2000_02-07-31_GMT.txt	197 kB	11/28/2014 5:35:12 PM
GP2A_Light_sensor_2_Jan_2000_02-07-31_GMT.txt	0 kB	11/28/2014 5:35:13 PM
GP2A_Proximity_sensor_2_Jan_2000_02-07-31_GMT.txt	0 kB	11/28/2014 5:35:13 PM
GpsLocation_2_Jan_2000_02-07-31_GMT.txt	0 kB	11/28/2014 5:35:12 PM
GpsNmea_2_Jan_2000_02-07-31_GMT.txt	0 kB	11/28/2014 5:35:12 PM
GpsStatus_2_Jan_2000_02-07-31_GMT.txt	0 kB	11/28/2014 5:35:12 PM
Gravity_Sensor_2_Jan_2000_02-07-31_GMT.txt	177 kB	11/28/2014 5:35:12 PM
Linear_Acceleration_Sensor_2_Jan_2000_02-07-31_GMT.txt	193 kB	11/28/2014 5:35:12 PM
MPL_Accelerometer_2_Jan_2000_02-07-31_GMT.txt	177 kB	11/28/2014 5:35:13 PM
MPL_Gravity_2_Jan_2000_02-07-31_GMT.txt	177 kB	11/28/2014 5:35:12 PM
MPL_Gyroscope_2_Jan_2000_02-07-31_GMT.txt	196 kB	11/28/2014 5:35:13 PM
MPL_Linear_Acceleration_2_Jan_2000_02-07-31_GMT.txt	193 kB	11/28/2014 5:35:12 PM
MPL_Magnetic_Field_2_Jan_2000_02-07-31_GMT.txt	186 kB	11/28/2014 5:35:13 PM
MPL_Orientation_2_Jan_2000_02-07-31_GMT.txt	177 kB	11/28/2014 5:35:13 PM
MPL_Rotation_Vector_2_Jan_2000_02-07-31_GMT.txt	192 kB	11/28/2014 5:35:12 PM
Orientation_Sensor_2_Jan_2000_02-07-31_GMT.txt	177 kB	11/28/2014 5:35:12 PM
Rotation_Vector_Sensor_2_Jan_2000_02-07-31_GMT.txt	198 kB	11/28/2014 5:35:12 PM

Figure 2-10: All recorded sensors on a classic smartphone using the Cellbot application on a Samsung Galaxy (c)

Many sensors are directly available, with two orientation fusion hardware already present (corresponding to MPL_orientation and Orientation_Sensor on Figure 2-10). Therefore for this technique we decided to use directly those data to estimate orientation, as they present no additional computational load, and are most likely taking account of some form of calibration. It should be noted that even if these considerations may seem very specific, most of the modern smartphones possess similar hardware inertial fusion implemented.

2.3.2 The detailed algorithm

With the context and global scheme of the technique explained, the detailed choices of implementation of the technique will be displayed.

2.3.2.1 Extension to in-plane rotation

As demonstrated in [Rublee et al. 2011] [Leutenegger et al. 2011], in-plane rotation invariance can be obtained through rotation of the pattern of a binary descriptor. Thanks to the use of inertial sensors, we have an estimation of the 3D rotation in Euler angles, noted $\alpha_t, \beta_t, \gamma_t$ being the yaw, pitch, and roll angles respectively measured at time t. Knowing the roll angle, we can rotate the couples (pos_i, pos'_i) around the position p that is described in order to obtain an in-plane invariance of the descriptors:

$$Rpos_i = R(p, \gamma_t) * pos_i \qquad (18)$$

Where $R(p, \gamma_t)$ is the rotation matrix describing the planar rotation of angle γ_t around the feature point p. We usually take the point p as the origin of the point's pair correspondences

position, and just add a translation to the point's pair rotated positions with the point p coordinates. This rotation operation has to be performed only once before describing every point, for every couple (pos_i, pos'_i). In the presented approach, we use a 256-bit long descriptor, leading to 256 rotations to compute compared to a classical binary descriptor strategy. This is a very reasonable cost compared to classical visual invariance extensions that need a visual clue on the feature orientation. As in-plane rotation does not affect the performance of the descriptor method, any angle can be handled with this strategy, achieving complete robustness to roll rotations.

2.3.2.2 Extension to viewpoint changes

As the BRIEF descriptor describes a planar area around a point with fixed point pair's comparisons, it is not robust to yaw and pitch rotations (viewpoint changes). It would be impossible to make the descriptor fully invariant to these transformations, as high rotations applied to the couples (pos_i, pos'_i) would lead to an overly slim shape of the described area, weakening its discriminating characteristics. For that reason, only small but still problematic angles changes were considered in this study, typically up to 20° for yaw and pitch angles. For roll however, as stated previously, no limitation has to be made.

To apply the rotation exactly, we need to convert the point pairs to 3D coordinates: $Xpos_i = (u_i, v_i, 0,1)^T$ where the coordinates of the points are $pos_i = (u_i, v_i)^T$. The inertial orientation is then converted to a rotation matrix R_t. As presented in section 1.4.3.1 the points $Xpos_i$ are rotated with R_t to points $X'pos_i$. Then a simple projection with a projection matrix corresponds to a projection of the points back onto a plane:

$$Rpos_i = \begin{pmatrix} Ru_i \\ Rv_i \\ Rw_i \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} X'pos_i \qquad (19)$$

Please note that any orientation representation can be used, it only has to be converted into a rotation matrix. With this strategy, the plane of point's pairs is rotated along the orientation measured by inertial sensors. The projection part only consists in keeping the three first coordinates of the point $X'pos_i$. Therefore the cost of computation is included in the rotation matrix calculus and application to the points, and the division of coordinates of the homogenous point $Rpos_i$.

2.4 Implementation and Results

In this section, results of the technique will be displayed to illustrate the gain realized. Implementation choices will be shown in order to reach the best performances possible with some approximations of the rotations computation. A quick computational cost analysis is then performed, to estimate the additional cost generated by the strategy.

2.4.1 Results

Two sequences will be detailed here to illustrate the improvement of the technique applied on the BRIEF descriptors. Note that it could also be applied to the BRISK [Leutenegger et al. 2011] or FREAK [Alahi et al. 2012] pattern. On all sequences, the FAST detector is used to extract interest points one the first frame. The BRIEF and Inertial-BRIEF (IBRIEF) techniques are applied to describe the points. Then the points of the first frame are matched with points for every frame. The first sequence, shown on Figure 2-11, mainly contains a roll motion to display in-plane rotation invariance. The second sequence, presented on Figure 2-12, contains several viewpoint changes.

Figure 2-11: A few frames extracted from the first sequence: we match the first frame (left) with other frames in the sequences (right). This sequence consists in a progressive roll motion.

Figure 2-12: A few frames extracted from the second sequence: we match the first frame (left) with other frames in the sequences (right). This sequence consists in multiple viewpoint changes.

2.4.1.1 Resilience to in-plane rotation

For the first sequence with every descriptor, an algorithm of perspective motion (1.4.2 estimation has been performed based on the matching provided by each descriptor type. It is based on the RANSAC (3.2.1 architecture that allows a classification of the matches: either inlier (good matches that fits the transformation) or outliers (bad matches that do not fit the transformation). A measure of the quality of the matches is the inlier percentage amongst the computed matches. We use this percentage in function of the frame number to display the efficiency of a descriptor. The result of the BRIEF [Calonder et al. 2010], ORB [Rublee et al. 2011], and presented IBRIEF are displayed on Figure 2-13. The implementations of BRIEF and ORB used in this experience were the ones available in OpenCV.

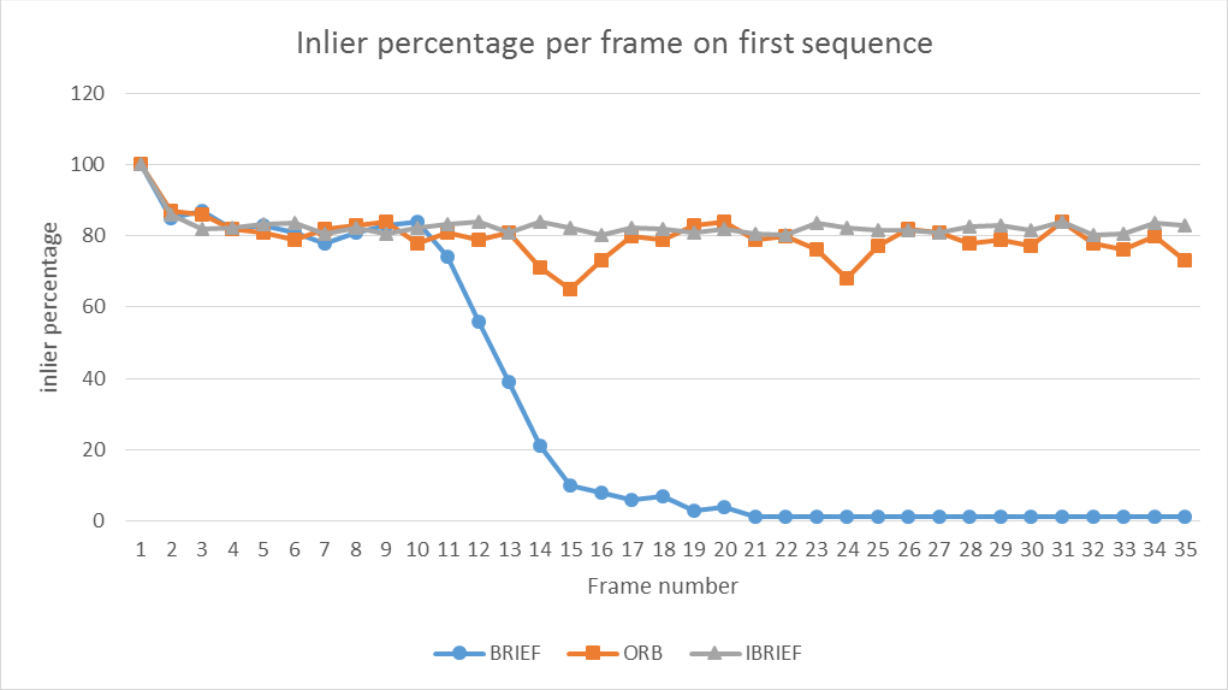


Figure 2-13: inlier percentage for three tested descriptors on sequence 1

For the BRIEF technique, that does not present invariance to in-plane rotation, one can see that the inlier percentage drops logically from a certain amount of rotation. The ORB algorithm is invariant to rotation and presents a more or less steady inlier percentage. However, two little decreases happen in frame 15 and 24. We believe that this is due to the fact that the BRIEF rotated grid is interpolated in 12 directions, and this frame may correspond to an orientation lying in the middle of two interpolated grids. Finally, the IBRIEF technique is the steadiest one in terms of inlier percentage.

2.4.1.2 Resilience to viewpoint changes

In the second sequence, multiple viewpoints changes occur. The sequence was recorded so that a book cover in the first frame appears on every frame. To illustrate the performance of the various methods, we display not only the inlier percentage (fig.2-13), that highly vary in the sequence, but also a warping of the first frame in the current frame perspective with the transformation found by the RANSAC algorithm. Figure 2-14 displays the results on the sequence for ORB, BRIEF, and IBRIEF.

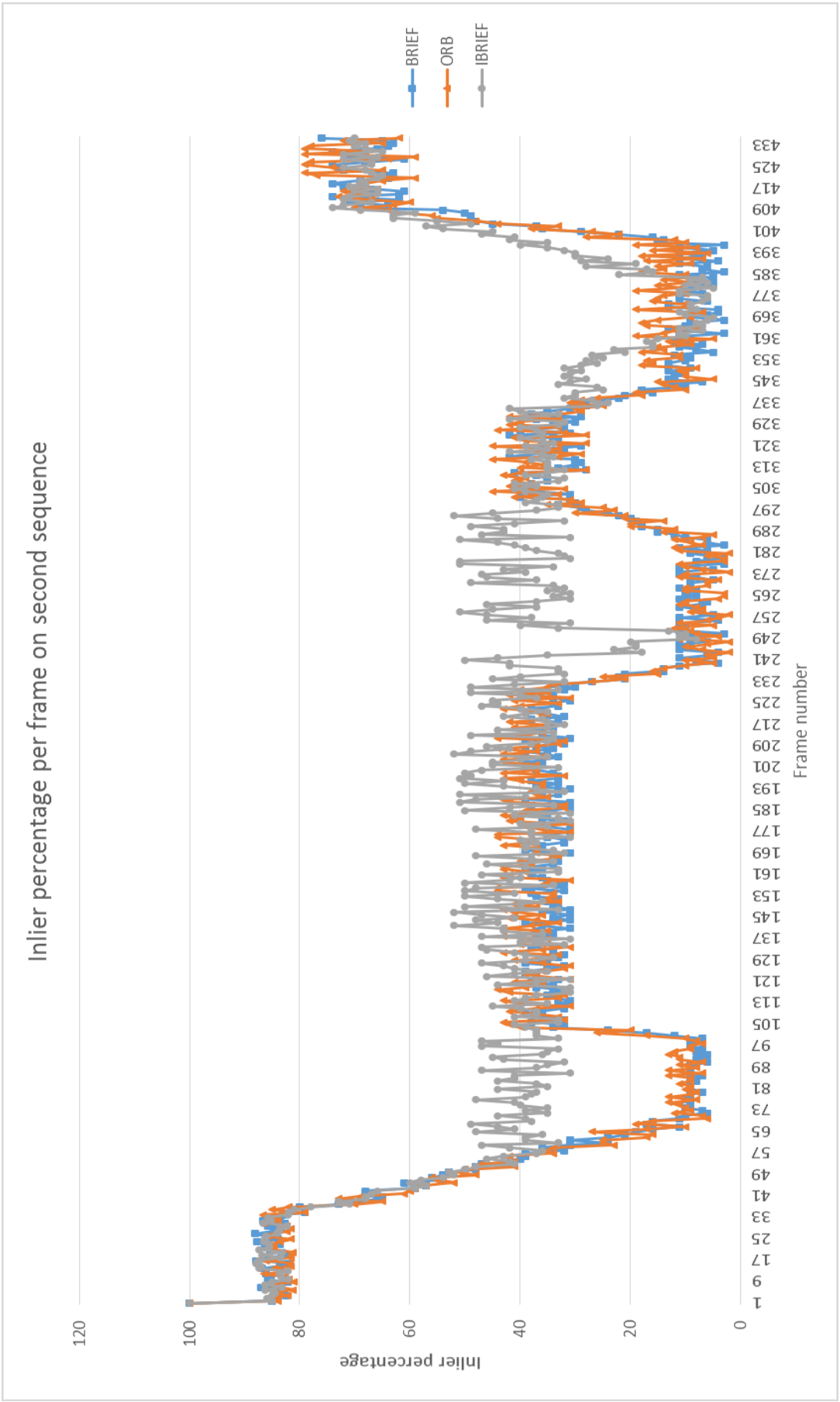


Figure 2-14: Inlier percentage per frame on the second sequence.

As expected, BRIEF and ORB perform in a similar manner, showing low robustness with regard to viewpoint changes. It was to be expected, as ORB does not add any improvement on top of BRIEF regarding invariance to viewpoint changes. IBRIEF clearly outperforms them when the rotation is high enough (frames 50-100, 233-290, and 337-400). However, when the rotation is even higher, even IBRIEF fails to reach correct performance.

On Figure 2-15, the inlier percentage in function of the dominant viewpoint angle is plotted. This confirms our hypothesis: a binary pattern rotation can help dealing with viewpoint changes, but when a certain amount of rotation is reached, the descriptor becomes too thin. In our experiments the yaw and pitch angles where even IBRIEF becomes unable to display good performance is ~25°. Some irregularities occur for smaller values, but a significant drop can be observed above this value.

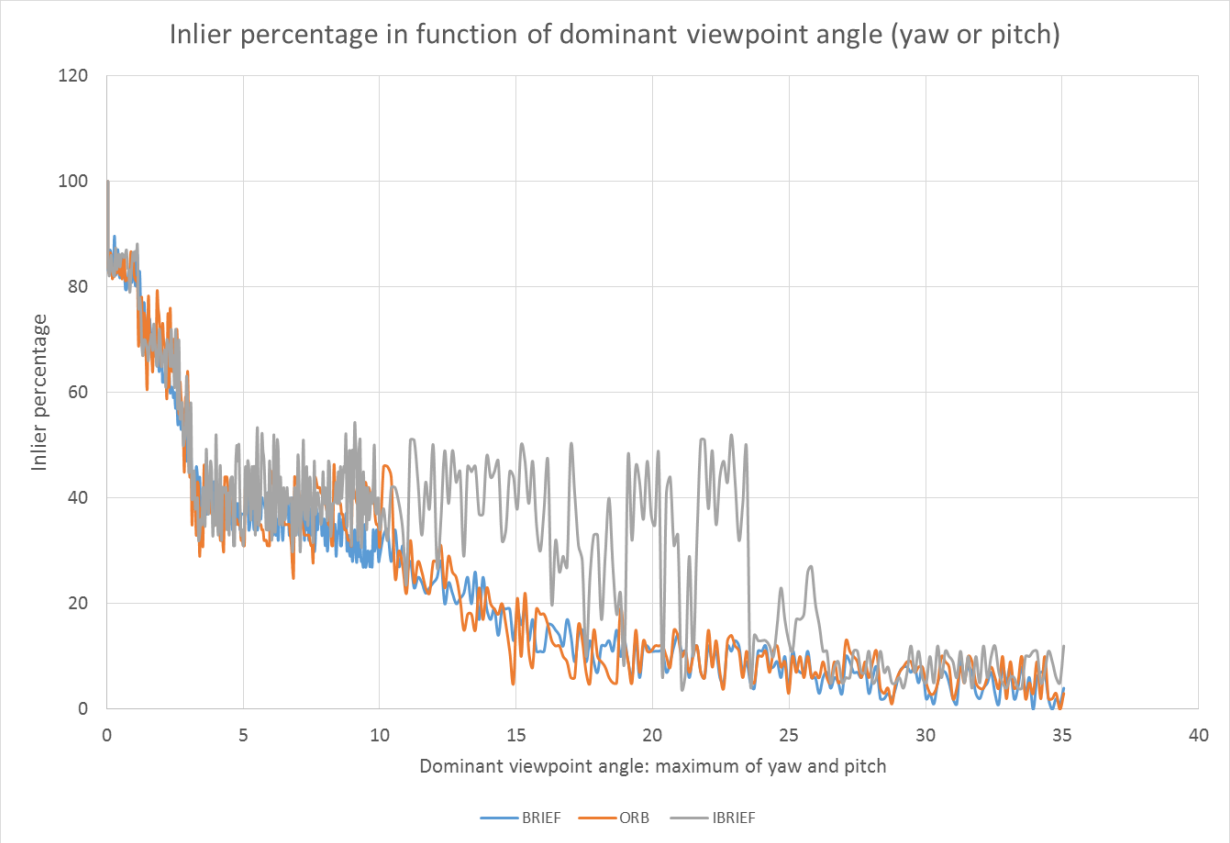


Figure 2-15: Inlier percentage in function of the highest viewpoint angle (max of yaw and pitch) for sequence 2.

In Figure 2-16, the first frame of the sequence is warped along the found perspective by the RANSAC algorithm. This shows that with high rotation angles, the method becomes ineffective.

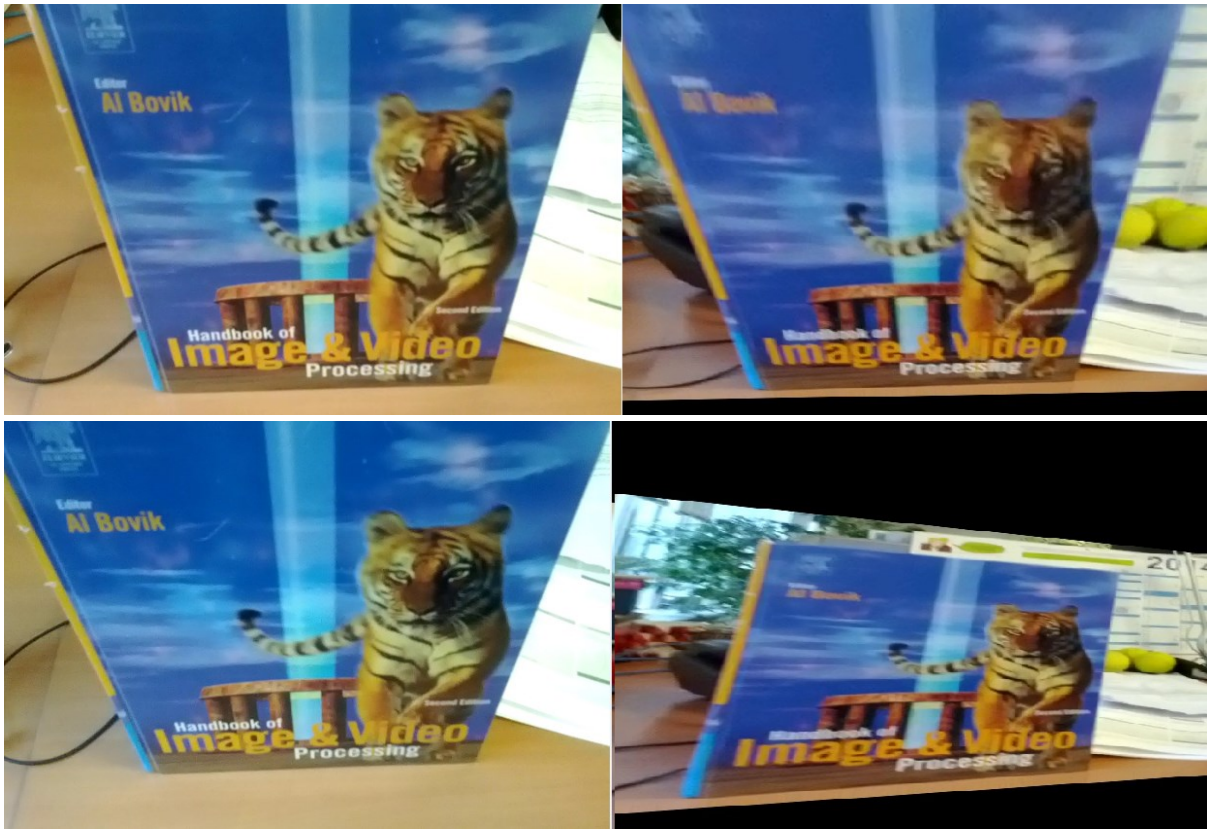


Figure 2-16: Two frames of the sequence (left) and the first frame warped in the perspective found by IBRIEF matches and RANSAC algorithm. In the first frame (353), the pitch is low-enough to allow a good performance by the technique. In the second one (375) the pitch is too high for the approach.

2.4.2 Implementation: Rotation approximation

The additional cost to the BRIEF technique in our strategy is the computation of the rotated pattern of the point's pairs. As seen in 2.3.2.2 , it consists in a rotation application to each point of the pattern, once per frame. In the context of this experience, the orientation is measured by the inertial sensors in Euler angles. We therefore need to convert these Euler angles into a rotation matrix in order to compute the rotation. Given the Euler angles $\alpha_t, \beta_t, \gamma_t$ that represent the rotation at time t, a rotation matrix that is generated by these angles is:

$$R_t = \begin{pmatrix} C_\alpha C_\gamma & S_\alpha S_\beta C_\gamma - C_\beta S_\gamma & C_\alpha S_\beta C_\gamma + S_\alpha S_\gamma \\ C_\alpha S_\gamma & S_\alpha S_\beta S_\gamma + C_\beta C_\gamma & C_\alpha S_\beta S_\gamma - S_\alpha C_\gamma \\ -S_\alpha & C_\alpha S_\beta & C_\alpha C_\beta \end{pmatrix} \qquad (20)$$

Where C_θ and S_θ are respectively the Cosine and Sine of angle θ. Please note that as stated in 1.4.3.2 , this depends on conventions, as the order of application of angles can differ. Applying this formula to our point's pairs gives us:

$$Ru_i = C_\alpha C_\gamma u_i + \left(S_\alpha S_\beta C_\gamma - C_\beta S_\gamma\right) v_i \qquad (21)$$

$$Rv_i = C_\alpha S_\gamma u_i + (S_\alpha S_\beta S_\gamma + C_\beta C_\gamma) v_i \quad (22)$$

As stated previously, the yaw and pitch angles will be limited in our approach, as they would make a flat descriptor when taken too high. Therefore we intended to approximate these formulas in case of low values for α_t, β_t. By using both suppression of terms with multiple sines products and small angles approximation, this leads us to:

$$Ru_i \approx (1 - \frac{\alpha^2}{2}) C_\gamma u_i + \left(-(1 - \frac{\beta^2}{2}) S_\gamma \right) v_i \quad (23)$$

$$Rv_i \approx (1 - \frac{\alpha^2}{2}) S_\gamma u_i + \left((1 - \frac{\beta^2}{2}) C_\gamma \right) v_i \quad (24)$$

To confirm that these approximations are valid in our application, a difference between the exact value of Ru_i and Rv_i is computed. The point's pairs in a BRIEF descriptor are located on a 32x32 pixels grid, which implies that the furthest point is located 16 pixels in each direction from the center. A 25° angle change in yaw or pitch only causes the difference between the accurate and approximation values to be 0.94 pixels for a point located in the corner of the patch, as one can see on Figure 2-17. Having the most impacted point by the approximation to be shifted by one pixel seem like a tolerable error for our application, as the location is always rounded in a BRIEF algorithm, that only uses point's pair comparisons.

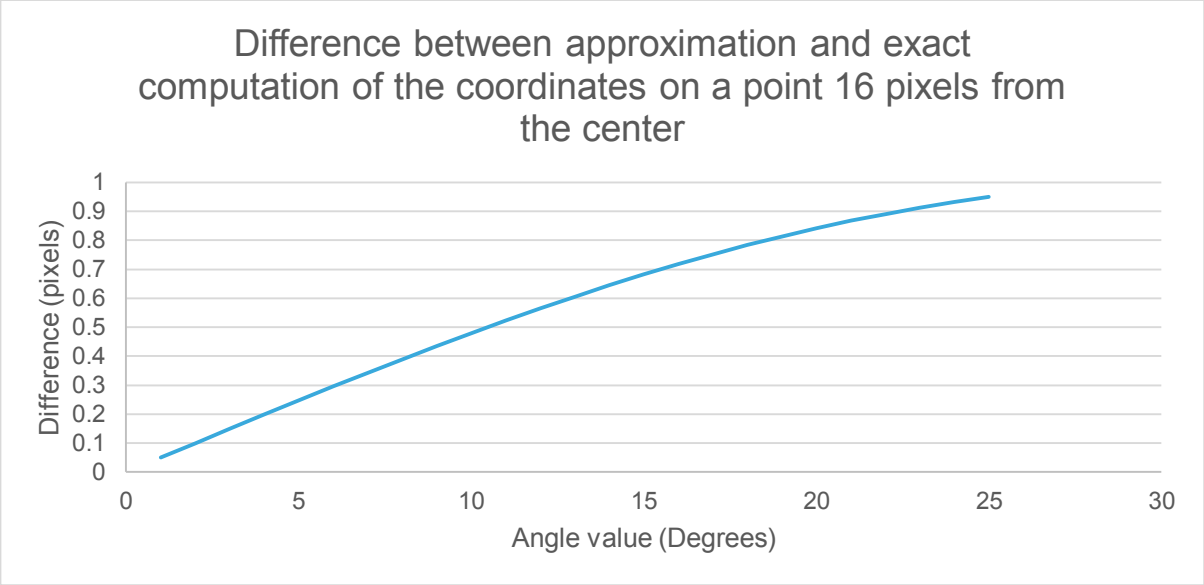


Figure 2-17: Difference between approximation and exact value of rotated points in the BRIEF pattern.

2.4.3 A quick word computational cost

In techniques such as ORB, the orientation of a feature is extracted from an intensity centroid that is found in the patch. This is performed by summing patches intensity weighted by position in the patch, to find its moments. Then a pre-computed rotated patch is picked. In our approach,

the computation load is not needed in the estimation of the orientation, as it is directly measured by the inertial sensors. However, as the viewpoints changes are also treated, pre-computing rotated patches would take too much space, as there is three dimensions of orientation. Therefore the computation overload to BRIEF is on the computation of rotated point's pairs.

The only additional cost to a traditional BRIEF of our method can be expressed as 512 (number of points pair in BRIEF) locations rotations as shown in equations (23) and (24). One great feature of this strategy is that this cost is not reliant on the number of points described. In effect, we only perform a modification to the pattern of BRIEF. Once computed, the new pattern is applied in a similar manner as BRIEF. Meanwhile, techniques based on visual information only require an estimation of the dominant orientation for every feature, scaling on the number of points.

2.5 Conclusion

A quick summary on the state of the art on local motion estimation was made. As in the context of the study mainly embedded platforms are targeted, we chose to focus on very light approaches, namely the FAST detector and BRIEF feature descriptors. An overview of hybrid keypoints shows that the addition of inertial sensors to those binary descriptors has not been shown in the literature. Inertial sensors can help the motion vector computation by estimating the orientation of the platform that can be used to increase robustness to geometric changes. However, inertial sensors do not help the visual techniques regarding scale estimation.

As motion blur invariance would require high accuracy on the motion estimation, MEMS inertial sensors are not suitable to reach this performance. Finally, IBRIEF is introduced, a technique that not only adds in-plane rotation invariance, but also improves the resilience to viewpoint changes of the BRIEF techniques. Tests show that it clearly outperforms classical descriptors. Our method has a fixed cost in terms of computation, and is a very light addition to the BRIEF algorithm.

Chapitre 3: Estimation de mouvement planaire hybride basée sur les vecteurs de mouvement

Below is a French summary of chapter 3: Hybrid planar motion estimation based on motion vectors.

Ce chapitre traite de la seconde étape des techniques classiques d'estimation de mouvement planaire, le calcul du mouvement global basé sur les vecteurs de mouvement. Les applications visées lors de cette thèse nécessitent un traitement robuste et rapide des données. Ceci donne lieu à deux contraintes majeures sur le système : la résilience vis-à-vis de vecteurs de mouvement incorrects, et la performance temps-réel du système.

L'intégration des mesures inertielles aux méthodes d'estimation planaire de la caméra peut se faire de deux façons. Tout d'abord le couplage fort, où les capteurs inertiels sont utilisés comme principale source d'information, la vidéo ne servant qu'à calibrer de façon temporelle et spatiale les capteurs. Ensuite, le couplage faible, où les deux estimations sont effectuées en parallèles, et le résultat final est donné comme une combinaison des évaluations du mouvement. Nous présentons un système qui n'appartient pas directement à une des deux catégories, car nous pensons pouvoir mieux tenir compte des différentes caractéristiques des capteurs en sortant de ces schémas de fusion.

Ce chapitre présente tout d'abord les différentes méthodes de régression robuste. Les techniques se basant sur la procédure « RANdom SAmple Consesus » sont étudiées de façon plus approfondie. Le nouvel algorithme proposé est présenté et détaillé. Enfin, des comparaisons avec d'autres méthodes et une étude de complexité calculatoire sont menées.

Chapter 3: Hybrid planar motion estimation based on motion vectors

This chapter focuses on the second step of classical camera motion estimation techniques, the estimation of a motion model based on local motion vectors. Two major requirements arise for a global camera motion estimator in our context: robustness to incorrect motion vectors and real-time performance.

Motion vector generation based on keypoint techniques often presents a lot of incorrect motion vectors, or outliers. Therefore, camera motion estimation requires robust methods to be applied on the motion vectors. A complex problem in those methods is the characterization of the noise in the data. Different scenes and motion types will lead to very variable constraints and effects on the motion vectors.

The scope of this thesis is embedded motion estimation. Thus, real-time performance must be reached, which requires for the global motion estimation algorithm to present a bounded and steady computational time. This constraints a lot the approach, as iterative strategies and robust regression techniques often have highly variable amount of computations.

The integration of inertial sensors in camera motion estimation is often proposed in two ways. Firstly, the loosely coupled fashion: the two estimations are performed independently and then recombined to propose a final estimation. Secondly, tightly coupled systems use inertial sensors as the principal source of information, and only use vision to calibrate the inertial sensors, both in terms of temporal synchronization as well as axis alignment. We will aim at performing a fusion that does not directly belong to the loosely or tight coupled categories. In effect, we believe that 2D camera motion estimation from inertial and visual sensors can benefit from deeper considerations of the respective estimations characteristics and drawbacks.

An overview of robust camera motion estimation techniques is proposed in section 3.1 . Classical regression methods, such as least-square are often applied on the motion vectors to find the global camera motion. To further increase the robustness of those approaches, M-estimators present numerous advantages. A short summary of visual-inertial fusion is also displayed in this section.

The RANdom SAmple Consensus (RANSAC) technique applied to global motion estimation and its variations are described in section 3.2 . In the modern literature, those types of algorithms have encountered a wide success, thanks to their robustness to incorrect motion vectors. In the RANSAC methods, datum is classified as correct (inliers) or incorrect (outliers)

in an explicit manner, which enhances the performance of these techniques compared to classical regressions methods.

The contribution of this thesis in the 2D camera motion estimation field relies on the hybrid RANSAC, which is detailed in section 3.3 . A novel manner of including the inertial measurement directly in the selection process of a preemptive RANSAC algorithm is introduced. Further quality improvements are made upon it.

Section 3.4 presents the results of the hybrid RANSAC, compared with classical algorithms. Computational time is also discussed. The hybrid RANSAC outperforms classical methods, without adding major computational steps to the process, therefore keeping a reasonable complexity.

3.1 Robust camera motion estimation

Many factors can create incorrect motion vectors for camera motion estimation ranging from a straight error of the technique (e.g. two different points with the same descriptor), to camera related issues (e.g. motion blur) and scene characteristics (e.g. internal motion, lack of texture). To compensate for this issue, many strategies exist. Some visual based methods intend to perform robust regression algorithms on the data to extract a correct motion model. Others use different means to perform motion estimation with inertial sensors to perform proprioception, which is not sensitive to the camera recording, but produces other issues. Finally, hybrid methods fuse both motion vectors and inertial readings to estimate robustly the camera motion. Figure 3-1 shows an example how a global motion estimation can be utilized to stabilize a video content.

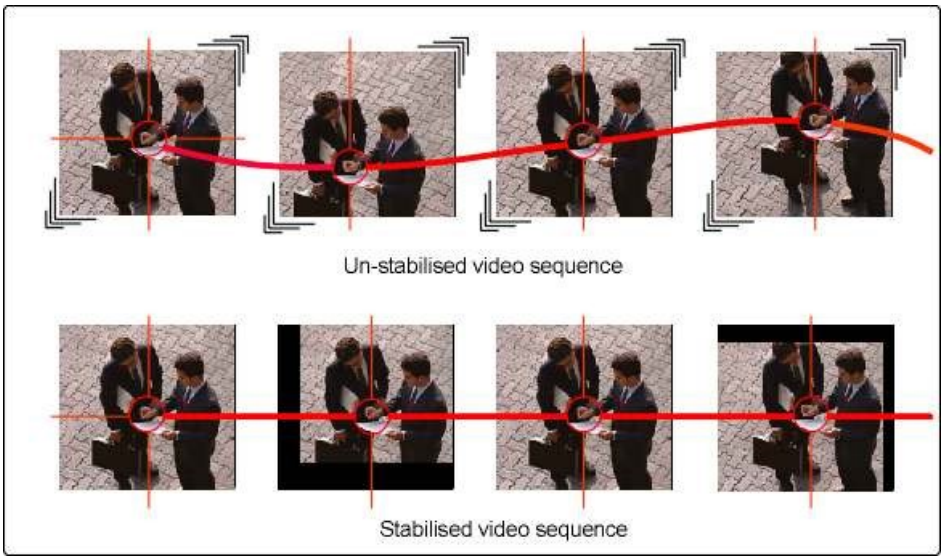


Figure 3-1: Example of an application to camera motion estimation: video stabilization. Extracted from [9]

[9] http://www.ovation.co.uk/video-stabilization.html

3.1.1 Robust regression techniques

Given the computed motion vectors, regression strategies can be used to compute a model that minimizes the errors regarding all the vectors. Diverse methods exist to perform such a task, the most well-known being the least-square method. Some variants built upon it make use of additional weighting terms or energy function to reach further robustness to noise.

3.1.1.1 Least square

We note h the 3x3 matrix corresponding to a motion model as presented in 1.4.2 . The motion vectors $v(i)$ are a sparse mapping of points coordinates from one frame to the next: $v(i) = (X_i^{prev}, X_i^{curr})$, where X_i^{prev} is the point on the previous frame and X_i^{curr} the point on the current frame. The least square approach minimizes the squared residuals ${r_i}^2$ of each motion vector, these are defined as:

$$ {r_i}^2 = \left\| X_i^{curr} - hX_i^{prev} \right\|_2 \qquad (25) $$

The goal of the least-square algorithm is to find the model $\hat{h}$ that minimizes the sum of the error terms:

$$ \hat{h} = \underset{h}{argmin} \sum_{i=0}^{N-1} {r_i}^2 \qquad (26) $$

This can be performed directly using techniques such as the Gauss-Jordan method. While this method is very efficient, it does not really cope well with general noise. In effect, if the noise on the data is assumed Gaussian then this estimation can present acceptable accuracy if enough motion vectors are provided. However, a non-symmetry in the noise on the data affects directly the estimation, especially if some vectors are very incorrect, leading to very high residuals that will predominate on other points.

3.1.1.2 M-estimators

To compensate for that, some strategies intent to minimize a function ρ of the residuals rather than the residuals directly. These are called M-estimators, as they belong to the maximum likelihood estimator class. The main benefit from using a function of the residuals is that it can lead to more robustness to noise. Many choices can be utilized for ρ, but the function should: always be positive, symmetric ($\rho(a) = \rho(-a)$) and strictly increase as $|r_i|$ grows. The goal is to limit the effect of a very high residual:

$$ \hat{h} = \underset{h}{argmin} \sum_{i=0}^{N-1} \rho(r_i) \qquad (27) $$

While general minimization of these terms can be complicated to perform, it becomes much easier when ρ is differentiable. If the estimator is differentiable, it is named ψ-type, while non-differentiable are named ρ-type. ψ is called the influence function. The solution to equation (32) now becomes:

$$\sum_{i=0}^{N-1} \psi(r_i)\partial r_i/\partial h_k = 0 \qquad (28)$$

for every parameter of h. These equations are equivalent because ρ is symmetric and strictly increasing from 0. Therefore the only value for which its differentiation is 0 is its minimum, which corresponds to a null residual. If we define a function $\omega(r_i) = \psi(r_i)/r_i$, then this system of equation can be related to a weighted least-square problem:

$$\hat{h} = arg\min_h \sum_{i=0}^{N-1} \omega(r_i){r_i}^2 \qquad (29)$$

This representation has the advantage to be solvable iteratively: we solve the equation (28), compute the term $\omega(r_i)$ with the new value of $\hat{h}$, and start the process again until a criterion is reached: the difference between two successive $\hat{h}$ is limited or a number of iterations have been processed. Many choices are available for ρ, with implications on the behavior of the method. Here are a few examples of functions that can be chosen:

- For L2 norm: $\rho(r_i) = \rho(r_i)^2/2 \ ; \psi(r_i) = r_i \ ; \omega(r_i) = 1$
- The Huber function changes around a threshold r_{lim}:

$$\begin{cases} if \ |r_i| < r_{lim} \\ if \ |r_i| > r_{lim} \end{cases}; \ \rho(r_i) = \begin{cases} {r_i}^2/2 \\ r_{lim}(|r_i| - r_{lim}/2) \end{cases}; \ \psi(r_i) = \begin{cases} r_i \\ r_{lim} sign(r_i) \end{cases}; \ \omega(r_i) = \begin{cases} 1 \\ r_{lim}/|r_i| \end{cases}$$

Other functions such as the Tukey bi-weight use function with a threshold that can be interpreted as a limit to the inlier / outlier separation. Figure 3-2 displays graphs of examples of the weight functions ω used for iteratively reweighted least-square techniques. More details on least square approaches can be found in [Fox 1997], [Ake 1990] and [Coakley 1997].

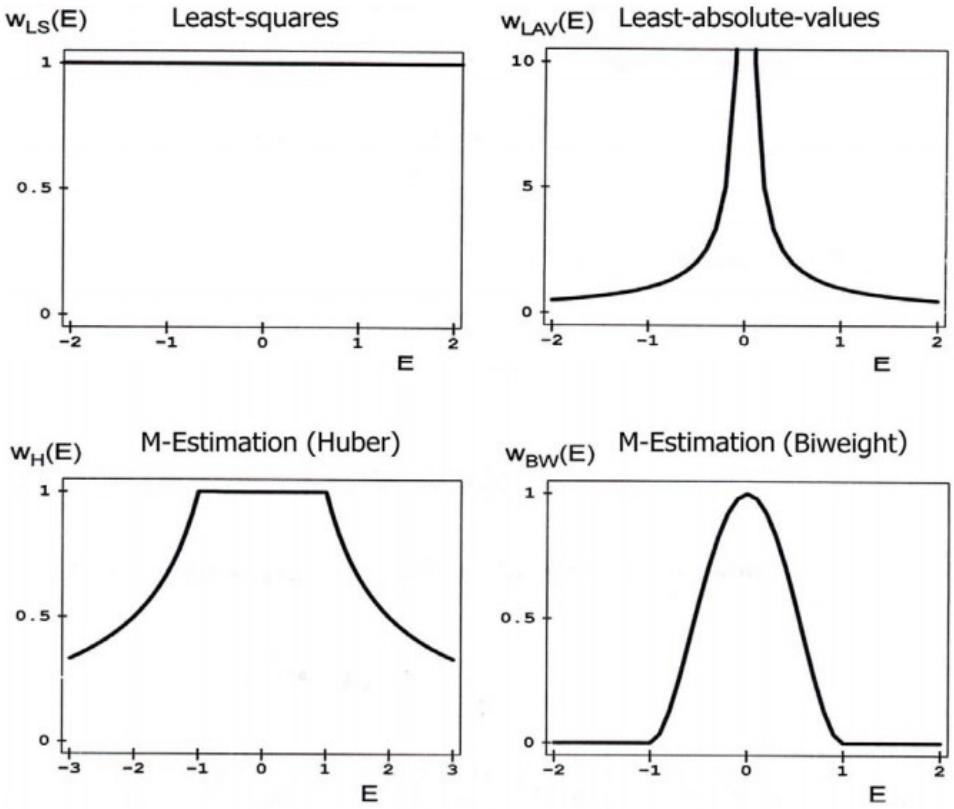


Figure 3-2: Function ω for several choices in M-estimators. Extracted from [Fox 1997]

3.1.1.3 Developed regression strategies of 2D motion estimation

Many motion estimations exhibit an architecture where a global camera motion is extracted from sparse correspondences, and a least-square or similar approach is utilized. Classical techniques of camera motion estimation were first introduced in [Bergen et al. 1992]. A Gauss-Jordan method applied to solve for motion model parameters is displayed in [Auberger & Miro 2005] while keeping a very low computational cost by removing outliers with specific techniques. In [Battiato et al. 2007], heuristics combined with two types of errors are used with the purpose of motion vector removal to avoid the effect of outliers. [Bergen et al. 1992] make use of a special cost function that approximates a L_p norm. To perform 3D video stabilization, [Liu et al. 2009] present a specific combination of energy terms that can directly be solved by a sparse least-square strategy. [Baker et al. 2010] utilize linear programming to solve a specific energy function that integrates an energy term that emphasis a smooth motion. A specific non-linear least square is proposed in [Forssen & Ringaby 2010] on LKT tracked points, in order to reduce rolling shutter distortions.

While an efficient way to solve the global motion camera estimation, these iterative regression techniques can be stuck in local minima, as they rely a lot on a good initialization and outlier noise models. They are also quite expensive to perform. This is why iterative "memory-less" methods, such as RANSAC or Least-median square where created. These will be detailed in section 3.2.1 .

3.1.2 Inertial and hybrid strategies

Inertial sensors possess very valuable characteristics for 2D embedded camera motion estimation applications, especially the gyroscope. In effect, it presents high frequency of measurement (typically 100-500Hz), and does not suffer from typical visual motion estimation issues, such as blur or internal motions. However, a few conditions need to be assessed to ensure the efficiency of the inertial sensors for this purpose, especially if the goal is to use them solely to compute the motion estimation. Calibration has to be performed, both temporally and spatially. Many inertial / hybrid approaches to camera motion estimation consider a rolling shutter camera, as most of the platforms of those applications have CMOS cameras.

Temporal calibration consists in synchronizing the camera recording with the inertial measurements. Spatial calibration is needed to correct eventual misalignments between the axis of the inertial sensors and cameras. [Karpenko et al. 2011] introduced a complete calibration process with a simple sequence, where the motion is estimated offline from SIFT features and RANSAC algorithm. Then the difference between inertial recording and visual estimations is minimized by findings parameters such as a multiplicative factor, static delay between estimations, and dynamic time scaling. The motion is then estimated directly with the inertial sensor readings and a specific motion model that uses line-by-line image warping.

Synchronization is performed as a parallel process of the motion estimation in [Hanning et al. 2011], where a few randomly selected points are tracked over frames in the video, then the delay between both sensors is computed based on the difference of both measurements. In parallel, an EKF in combination with a form of complementary filtering is performed on the inertial data, to fuse the gyroscope and accelerometer. Here, the inertial sensors are the main source of estimation, with visual information only used to calibrate the inertial and visual sensors.

Rather than using solely inertial sensors, some methods fuse the visual and inertial data to estimate the camera motion. In [Zhu et al. 1998], The camera motion is calculated with block matching techniques on points located on a fixed grid. With the camera motion estimated, a filtering is performed to stabilize the video content. The filter is based on the inertial sensors readings, which are used to determine the motions that should be preserved (intentional) and the ones that should be filtered (jittering motions). In this system, the fusion is not directly made to estimate the motion but rather to obtain information regarding the physical behavior of the platform.

[Jia & Evans 2012] introduced a method based on the Extended Kalman Filter (EKF), where specific approximations are made on the EKF that correspond to the video stabilization and rolling shutter correction context. In effect, they only consider 3D rotation as the motion, allowing reduced forms on the Jacobian computation. Figure 3-3 displays a rolling shutter distortion correction with this method. Inertial sensors measurements are integrated in the filter

as inputs in the dynamic model update. Each gyroscope reading is incorporated in the motion model, until an image frame is recorded. Then, points tracked with the LKT are used in the filter as measurements, in the correction step of the EKF. This type of fusion in a probabilistic filter to estimate the motion is widespread in the SLAM domain, where the state of the filter also incorporates 3D point's locations.

Figure 3-3: A picture captured with a CMOS sensor (left) and the rolling shutter corrected frame with the method of [Jia & Evans 2012]

In a similar manner, [Klein & Drummond 2004] present a system where both visual and inertial information are used in a probabilistic filter with an application to tracking. Inertial measurements provide rotation prediction, as well as a pre-estimation of the amount of blur that will occur in the frame, which is applied as a parameter in the visual edge detection. Then filtering is performed using visual measurements mainly.

3.2 Ransac and variations

Regression strategies are an efficient manner to compute a 2D motion model based on local motion vectors. However, it underperforms when too many outliers occur in the dataset. To avoid such issues, "memory-less" (the models from one iteration to the next are not related) iterative methods were created. They consist in creating a motion model based on a minimal number of pieces of data, and perform selection based on a criterion. They have encountered a lot of success, and many variations were created from them. The most famous of these algorithms is the RANdom SAmple Consensus (RANSAC).

3.2.1 Classical RANSAC

Introduced in [Fischler & Bolles 1981], The RANSAC algorithm is an iterative procedure that finds, among a set of proposed models, the one with the largest consensus, meaning the model that accounts for pieces of data with an error inferior to a threshold T. This model is computed with a confidence η, a parameter that has to be set by the user. The key of the method is to explicitly integrate the proportion of good pieces of data, or the inlier percentage ε, in order to determine the number of iterations needed to reach a confidence level of η that the best model

found is indeed the correct one. As it is applied in our context to motion estimation, the notation previously proposed is maintained: the input data is a set of N motion vectors $v(i) = (X_i^{prev}, X_i^{curr})$, where X_i^{prev} is the point on the previous frame and X_i^{curr} the point on the current frame and h is a 2D motion model in the form of a 3x3 matrix as shown in 1.4.2 .

3.2.1.1 The algorithm

a) The first stage of the procedure is to compute a model h_0 with a minimal amount m of randomly selected motion vectors $v(i)$ for the desired model type. For instance, a similarity transformation that incorporates in-plane rotation, scaling and translation requires 3 motion vectors.

b) In a second step, the created motion model is tested with all pieces of data available. For each motion vector the reprojection error is computed applying the proposed model to the previous points:

$$\epsilon\big(h_0, v(i)\big) = \left\| h_0 X_i^{prev} - X_i^{curr} \right\| \tag{30}$$

If $\epsilon\big(h_0, v(i)\big)$ is below the threshold T, the vector $v(i)$ is considered an inlier with respect to model h_0, otherwise it is an outlier for this model.

c) Thirdly, the total number of inliers k_0 leads to the inlier percentage for this model: $\varepsilon_0 = k_0/N$. This allow us to calculate the remaining number of iterations. In effect, with this estimated percentage of inliers, the probability to create a model with inliers only is ${\varepsilon_0}^m$. If the procedure is iterated K times, the probability that not a single model was created with inliers only is: $(1 - {\varepsilon_0}^m)^K$. As the desired confidence is η, the probability $(1 - {\varepsilon_0}^m)^K$ needs to be equal to $(1 - \eta)$:

$$(1 - {\varepsilon_0}^m)^K = (1 - \eta) \tag{31}$$

As we look for the number of iterations that should be produced:

$$K \geq \frac{log(1 - \eta)}{log(1 - {\varepsilon_0}^m)} \tag{32}$$

d) Fourthly, steps a) and b) are repeated K times, or until a better model h_j is found, in the sense that it displays a higher inlier percentage. If this occurs, then K is recomputed with the new inlier percentage, which leads to a lower amount of remaining iterations. Figure 3-4 displays the amount of iterations needed for a 0.999 confidence rate.

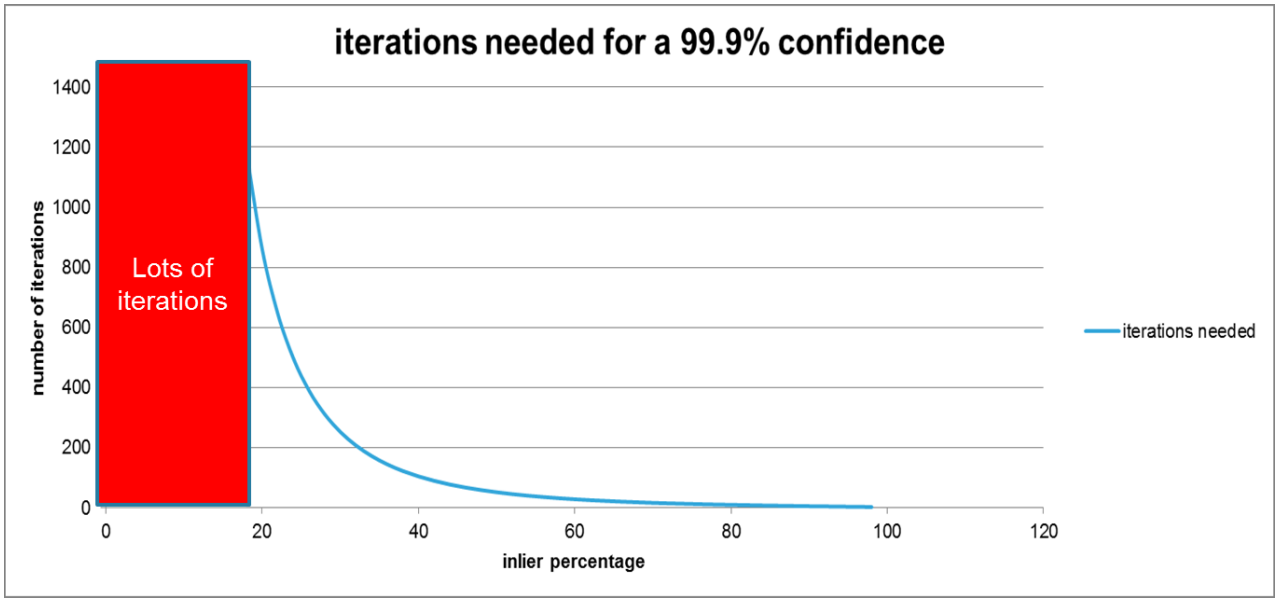


Figure 3-4: number of iterations needed in the RANSAC algorithm for a 99.9% confidence

e) Finally, when the number K of iterations has been reached, the best model is considered to be the one with the highest support, which corresponds to the highest inlier percentage. In most implementations, K is bounded so that the procedure does not produce too much computations. In some implementations, the best model h_{best} is recomputed with a least-square approach with all its inliers.

3.2.1.2 Characteristics

This procedure is "memory-less" in the sense that, for each iteration, the new model under testing is not dependent on the previous ones, only the amount of iterations relies on what was computed before. Other techniques, such as Least-Median-of-Square (LMedS) proceed similarly, but differ in considering the median error of a model h_j as the criteria of performance of a model, rather than its inlier percentage ε_j.

The main advantage of the RANSAC strategy is that it can extract a motion model without restriction on the amount / type of noise. It does not require massive parameterization, as only T and η have to be set. Thanks to the lack of dependency between two iterations, it is very prone to parallel implementation. However, this procedure cost can vary a lot, as it relies on the quality and quantity of the dataset.

3.2.2 Variations from RANSAC

To overcome the limitations of the classical RANSAC approach, methods were designed to improve it. The main goals of these improvements are to perform smarter testing through the data, to improve the generation of hypothesis and to apply optimization to the model while testing it.

[Torr & Zisserman 2000] adapted the RANSAC procedure to make its cost function correspond to an M-estimator with the Huber cost function. The inlier / outlier split is treated with expectation minimization technique, where a datum is not strictly classified, but rather the probability of it to be an inlier is computed. This allows more accuracy in the estimated transformations. This inlier probability per datum is also used to sort the data, allowing guided sampling approach that was introduced in [Tordoff & Murray 2002], where the tracking of multiple motions is performed.

To reduce the amount of computation spent on bad models, [Matas & Chum 2004] introduced a pre-testing procedure. This consists in checking a generated model with a reduced set of pieces of data; if the testing is successful then the model is tested with every point, otherwise it is discarded. In experiments, it was demonstrated that the optimal technique would be to test only one piece of data as the pre-test [Matas & Chum 2004]. In a similar goal, [Capel 2005] presented a bail-out test that consists in stopping the test procedure if a model do not seem to present enough inliers with a reduced set of data. A further interpretation of the data was developed in [Matas & Chum 2005; Chum & Matas 2008] where a consistent rate of outlier is explicitly modeled. This allows to determine a sequential probability ratio test that indicates in an optimal manner if a model should be rejected early.

Model generation is improved in the PROSAC technique [Chum & Matas 2005]. Pieces of data are first classified according to a similarity function (for instance the matching score for motion vectors). Then hypotheses are generated preferably with the pieces of data that present the highest confidences, progressively increasing the subset of data used to generate the models as the number of iterations increases.

Finally, a constant (or at least majored) RANSAC is presented by [Nistér 2005] called the preemptive RANSAC, which will be detailed in next section. [Raguram et al. 2008] introduced an overview of the RANSAC variations, using some of them in their ARRSAC procedure.

3.2.3 Real-time RANSAC: preemptive procedure

As detailed previously, the main drawback of the RANSAC technique is its very variable computational time that scales linearly with the amount of data and also relies heavily on the dataset quality, by the rate of inliers. To avoid those drawbacks, the RANSAC technique was adapted in a preemptive procedure in [Nistér 2005]. Here is an overview of the method:

1. Generate M motion models h_j, by selecting randomly a minimal number of vectors $v(i)$ to create each model. Initialize a score $S_j = 0$ for each model. Set the increment step $\mathrm{i} = 1$
2. For every model h_j, compute the reprojection error $\epsilon(h(j), v(i))$ on the set of vectors. If it is below the predefined threshold T for the model h_j, then update its score: $S_j = S_j + 1$
3. Only keep the best $f(i)$ models (sorting by score), where $f()$ is a preemption function.

4. If $f(i) \leq 1$ (only one model left) or $i = N$ (we have tested every motion vectors), keep the model with the highest score as the global estimation. Otherwise, set $i = i + 1$ and go to step 2.

The preemption function $f(i)$ defines the number of models kept at each stage of the algorithm. For instance, the one used in [Nistér 2005] is:

$$f(i) = \lfloor M2^{-\lfloor i/B \rfloor} \rfloor \quad (33)$$

Where B is a bundle size of motion vectors, and $\lfloor . \rfloor$ denotes the downward truncation function. In this example, we just divide by two the number of models considered after each bundle of B motion vectors.

The preemptive RANSAC allows having a limited computational time, regardless of the number of motion vectors, and of the quality of the dataset. As some motion models are deleted early and not tested with all the data, we save some computational time compared to a classical RANSAC.
This is done at the cost of losing the adaptability to the dataset quality of the classical RANSAC scheme, because the inlier percentage no longer influences the number of tested models (which is fixed in the preemptive approach). Figure 3-5 displays the differences in execution flow between a classical and preemptive approach.

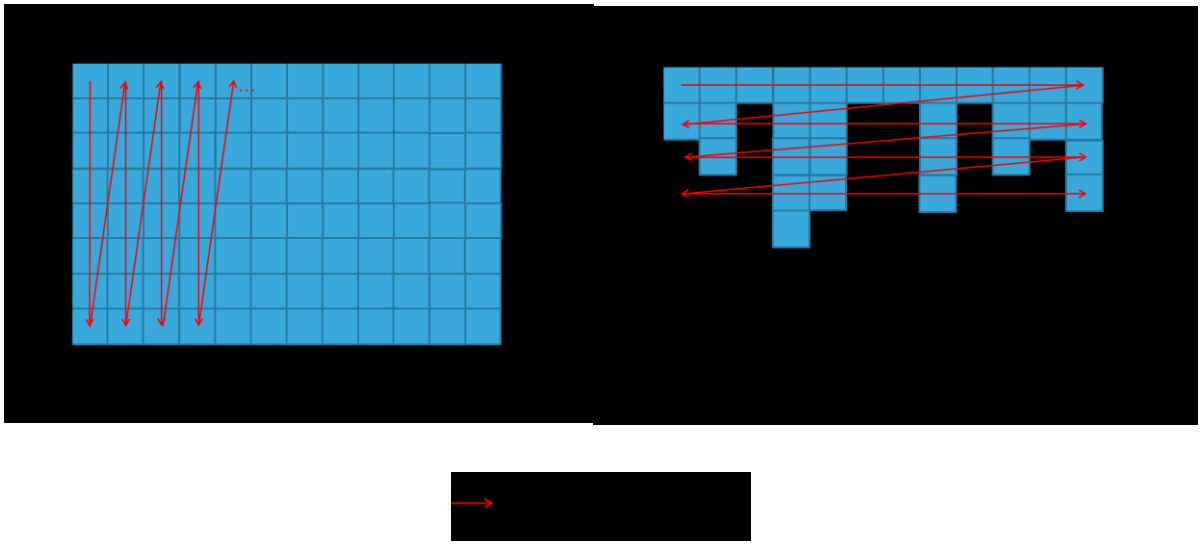


Figure 3-5: Execution flow difference between a classic and preemptive RANSAC approach

3.3 Hybrid Ransac

Visual-inertial sensor fusion has been popular in domains such as SLAM. It has also been used to solve motion artifacts [Zhu et al. 1998] [Jia & Evans 2012], to circumvent the failure cases of vision-based motion estimation as well as the heavy calibration processes of the inertial sensors-based estimation [Karpenko et al. 2011]. However these algorithms use probabilistic

filters to fuse data and estimate the motion. This implies tracking features on many frames, leading to error accumulation in case of false matching, possibly corrupting a full sequence for a one-frame mistake. This is why we created a frame-to-frame, real-time, online fusion algorithm, built upon the preemptive RANSAC scheme, to perform accurate and robust motion estimation [Alibay et al. 2014]. Figure 3-6 displays a scheme of the procedure of the hybrid RANSAC. The complete algorithm will be detailed in the next sections.

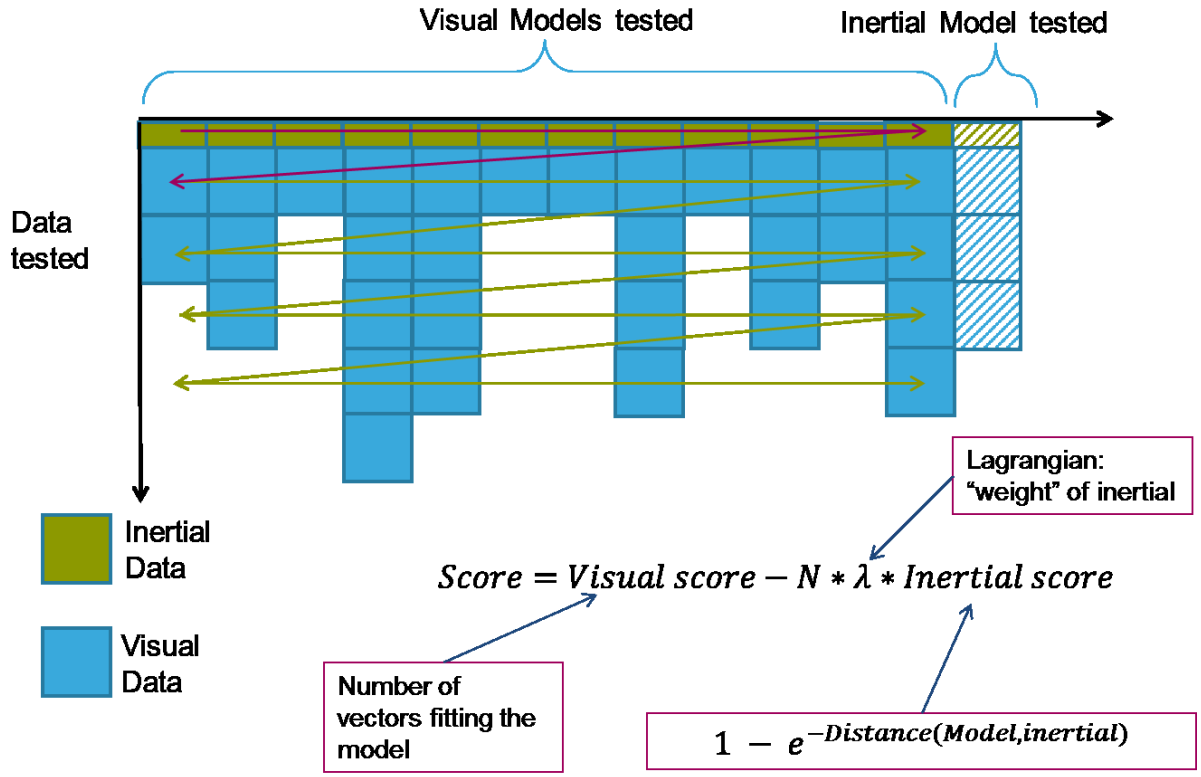


Figure 3-6: A scheme of the hybrid RANSAC procedure.

3.3.1 Hybrid scoring of the models

The RANSAC algorithm scores a candidate motion model through the number of motion vectors fitting this model: the more vectors fit, the better the model is considered. But when it comes to difficult sequences (for instance with an object passing in front of the camera), the desired best model does not always correspond to this definition, as we want to estimate the camera motion and not necessarily the motion visible in the scene. This is even more critical with a computation-bounded algorithm like the preemptive RANSAC, because some valid models can be rejected early in the process.

To overcome those drawbacks, inertial sensors are a valuable addition to the technique, as they provide information on the camera motion unrelated with the scene. Rather than combining the two motion estimations at the end of a process or in a filter, we designed a technique so that the inertial measurements have an active role in the model selection process of the hybrid RANSAC based on visual motion vectors.

In a preemptive RANSAC procedure, all models are created at the beginning of the procedure, then progressively selected based on their score S_j, which is computed by testing all the models with each motion vector. The main issue of this strategy is that the first motion vectors tested have a very high impact on the selection process, as incorrect motion vectors can lead to the early deletion of every suitable motion model.

That is why we transform the pure visual score S_j into a hybrid score S_j^{hyb}, damping the previous visual score with an inertial-related one:

$$S_j^{hyb} = S_j - N\lambda(1 - e^{-\partial\left(h_j, I(t)\right)}) \tag{34}$$

where λ is a lagrangian factor and $\partial(h_j, I(t))$ an error measure between the considered motion model h_j, and the inertial measurements $I(t)$, defined by the relative motion between the two frames measured by the inertial sensors. The computation of $\partial(h_j, I(t))$ relies heavily on the type of model that was chosen for the technique. For instance, with Euler angles and a similarity transformation, the computation can be explicated as:

$$\partial\left(h_j, I(t)\right) = \frac{\left(\frac{T_x}{\rho_x} - \alpha_t\right)^2 + \left(\frac{T_y}{\rho_y} - \beta_t\right)^2 + (\theta - \gamma_t)^2}{\partial_{carac}} \tag{35}$$

With $\alpha_t, \beta_t, \gamma_t$ being the yaw, pitch and roll angle displacements. T_x, T_y and θ are respectively the translation amounts and in-plane rotation angles extracted from h_j. ∂_{carac} is a normalization factor. As the preemptive procedure unfolds, models are selected based on their compliance with both visual and inertial measurements, thanks to the hybrid scoring.

3.3.2 Inertial model inclusion

The previously stated technique with the hybrid scoring of the model allows a better selection of the motion models, especially in the early stages of the preemptive RANSAC procedure. In some particular cases, improving the selection is not enough to reach an acceptable level of performance of the camera motion estimation.

As shown in Figure 3-7, some scenes show too much complexity for the camera motion to be estimated based on visually generated models. The first one displays too much internal motion, the second one have too much blur, while the third one is not textured enough to present satisfying results.

Figure 3-7: Some examples of very complex scene. Black vectors are the outliers computed by our method, while white vectors are the inliers.

All of those observations lead to a conclusion: an integration of the inertial measurements in the model generation process is also needed.

To cope with these extreme cases, we add a motion model purely created from $I(t)$, expressed as an affine transformation $h_{inertial}$. In the case of similarity transformation, we create the inertial model as:

$$h_{inertial} = \begin{pmatrix} \cos(\gamma_t) & \sin(\gamma_t) & \alpha_t \rho_x \\ -\sin(\gamma_t) & \cos(\gamma_t) & \beta_t \rho_y \\ 0 & 0 & 1 \end{pmatrix} \quad (36)$$

This transformation will be picked over the visual ones in cases where the visual motion fields fail at describing the camera motion, as the model $h_{inertial}$ is scored in the exact same manner as he other models h_j.

3.3.3 Dynamic lagrangian computation

The lagrangian value λ of the scoring equation (39) is very difficult to set for a given sequence. Indeed, having a high value will often favor the selection of the inertial model, leading to the inaccuracies and drifts of the inertial sensors to become apparent in the motion estimation. When performing applications such as video stabilization, this noise can even deteriorate the sequence rather than improving its quality, which is not acceptable.

On the other end, a low lagrangian value brings back the impact of visual failure cases (big internal motion, lack of texture), while the goal of the technique is to increase the robustness to these cases. Therefore it appears that a constant lagrangian factor is not suitable to reach good performance level.

A way to set λ dynamically should be found as we want to adapt the procedure to the difficulty of the motion field in real-time. The most difficult the scene, the higher the lagrangian should be. Another important requirement is that λ should be set before testing the motion vectors, because we need it to compute the hybrid score of each model.

To do so, we compute ∂_{med}, the median value of all the distances of visual models to the inertial one $\partial\left(h_j, I(t)\right), j = 1 \dots M$. The chosen measure is a reliable indicator of the difficulty of the motion vector field, since it computes the disparity in the inertial-visual distances. These will be high in case of internal motion, lack of texture and blur (where we want to favor the inertial model), but low otherwise. We compute the dynamic lagrangian λ_{dyn} as:

$$\lambda_{dyn} = \lambda_{max}(1 - e^{-(\partial_{med}/\partial_{carac})^2}) \quad (37)$$

where ∂_{carac} is a normalization factor, and λ_{max} the maximum lagrangian to be utilized. The main advantage of this parameterization is that its cost of computation is very light, while improving significantly the flexibility of the method.

3.3.4 Global procedure

We can now summarize the hybrid preemptive RANSAC algorithm with all the steps involved, as follows:

Algorithm: Hybrid Preemptive RANSAC

1. *Generate* $M-1$ *motion models* h_j*, by selecting randomly a minimal number of vectors to create each model. Create the last model using only inertial measurements as in (36)*
2. *Compute the distance between the visual and inertial models* $\partial\left(h_j, I(t)\right)$ *for each model as in (35)*
3. *Using the median value* ∂_{med}*, compute* λ_{dyn} *as shown in (37)*
4. *Initialize the hybrid score for each model* $S_{hyb}(j) = -N\lambda_{dyn}(1 - e^{-\partial(h_j, I(t))})$*. Set the increment step* $i = 1$
5. *Compute the reprojection error* $\epsilon(h_j, v(i))$ *for every model* $h(j)$*. If it is below* T*, then update the score of the model* h_j *as:* $S_j^{hyb} = S_j^{hyb} + 1$
6. *Only keep the best* $f(i)$ *models (sorting by score), where* $f()$ *is a preemption function.*
7. *If* $f(i) \leq 1$ *or* $i = N$*, keep the model with the highest score as the global estimation. Otherwise, set* $i = i + 1$ *and go to step 5.*

3.3.5 Online temporal calibration

One major requirement for the algorithm to perform efficiently is to maintain a correct synchronization between the visual data and inertial measurements. As the device can suffer from software latencies, or the various clocks can display relative drift between each other, the calibration needs to be maintained online to maintain a proper level of performance.

3.4 Results & discussions

The hybrid RANSAC strategy has been evaluated both in terms of computational time and quality performance against classical frame-to-frame correction techniques.

3.4.1 Testing protocol and results

Evaluating a motion estimation algorithm is never an easy task, due to the lack of ground truth available, especially when sequences are recorded on embedded platforms, and not virtually generated. A large database of sequences was recorded, including high internal motion (passing cars, moving objects), texture-less sequences (skies, walls) and high motion sequences. Three

algorithms were tested on the database: the pure visual preemptive RANSAC, online correction with inertial sensors only, and the proposed hybrid RANSAC. The camera motion was corrected in each sequence using the three methods, in order to compare subjectively the efficiency of each algorithm.

We found major improvements in terms of robustness compared to the pure visual preemptive RANSAC. On Figure 3-88, we plot the horizontal translation estimated by the three methods on a sequence of moving cars and trucks, where the camera has very little motion. We can see that the visual algorithm is trapped into internal motions, and that the inertial estimation is very noisy. Thanks to the hybrid scoring, our algorithm converges on a correct model.

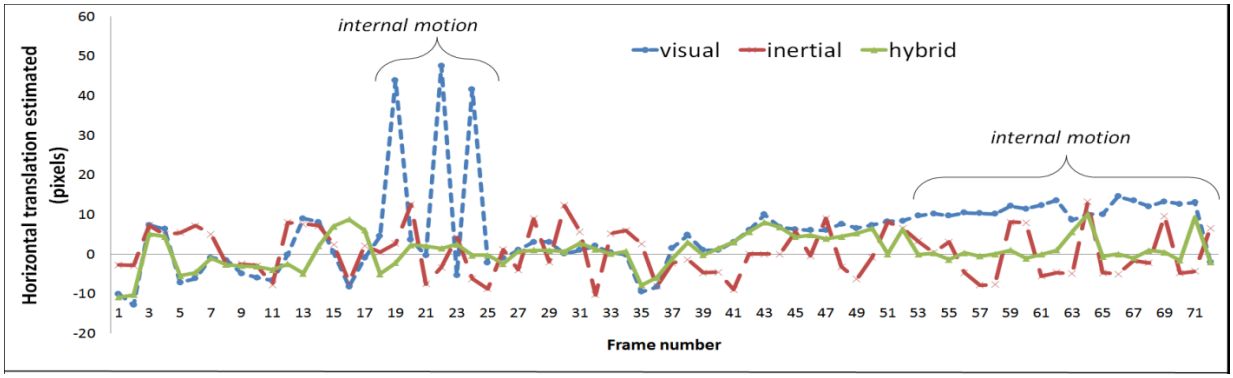


Figure 3-8: Estimation of the horizontal translation in a 120 frames long sequence. Three estimations are made: a visual one (preemptive RANSAC), an inertial one (compensation of gyroscopes input), and our proposed hybrid algorithm (hybrid RANSAC).

On multiple sequences (see Figure 3-99), our algorithm demonstrate superior performance to classical frame-to-frame approaches. The inliers (white vectors) and the outliers (black vectors) are represented on the frames. On the left, the estimated motion by a classic preemptive RANSAC algorithm, on the right the hybrid RANSAC algorithm. Blue vectors are the recorded inertial measurements converted into motion vectors, green vectors are accumulated inertial measurements over one frame. As the camera was a CMOS rolling shutter one, every line is recorded at different time, and a time of record can be put in correspondence with a line in the frame. The red vector on the middle of the frame displays the total motion estimated by the inertial sensors, which is computed by averaging the accumulated inertial vectors on the frame.

The first two frames display resilience to internal motions. In the first frame, the visual technique estimates the motion of the moving truck (all inliers on it) relatively to the camera which is very problematic as we want to estimate the motion of the camera only. The same issue occurs on the second frame with the paper moving in front of the camera. As the visual preemptive RANSAC estimates the best motion model as the one with the most inliers, and if most motion vectors belong to a moving object, it fails to recover the proper camera motion. On the other hand, the score of the hybrid RANSAC is a mix between inertial and visual measurements of the motion models. It is therefore able to correctly favor the motion model corresponding to the camera, even if it has less visual inliers than another model.

The third frame show how the inertial model included in the technique can help to circumvent very complex scene, where pure visual estimation cannot perform correctly. There are too few vectors in this texture-less scene. Thanks to the pure inertial model, the hybrid RANSAC is able to recover plausible motion model in these very complex cases, where it is very tough to recover any valuable information from visual cues only.

Figure 3-9: A few examples of test sequences by the preemptive RANSAC (left) and presented Hybrid RANSAC (right)

3.4.2 Computational time

One of the main requirements of the presented algorithm is to run online in real-time. With added computations due to the integration of inertial measurements, we want to evaluate if we are able to reach an acceptable time of execution. Using the same preemption function as in

[Nistér 2005], we can evaluate the complexity of our method. The computational time is approximately:

$$t_{exec} \sim M(t_m + \boldsymbol{t_\partial} + \boldsymbol{t_s}) + \frac{MB}{2} t_\epsilon \qquad (38)$$

with M being the number of motion models, B the number of vectors in a block, t_m the time to generate a model, $\boldsymbol{t_\partial}$ the time to compute each hybrid distance and score, t_ϵ the time to compute a reprojection error, and $\boldsymbol{t_s}$ the average time per model to find the median value (noting that finding the median value in an unsorted list of M values is proportional to M on average through a partial sorting procedure, we write it as $M\boldsymbol{t_s}$). A strong point of our approach is that we keep a bounded computational time, only adding the terms in bold face in Eq. (43) to the ones of the pure visual preemptive RANSAC. The overall complexity remains linear to the number of models. Therefore we manage to keep a real-time performance.
Performances of a direct implementation with no specific algorithmic optimization were measured on a 2.8GHz double core, 4GB RAM computer. The average time spent by frame by the preemptive RANSAC and the hybrid one are presented on Table 1, with $B = 30$. The hybrid RANSAC adds some computational time, but still reaches real-time performance (>30fps).

Number of models used	Pure visual preemptive RANSAC	Hybrid RANSAC
$M = 100$	5.07 ms	8.65 ms
$M = 200$	15.36 ms	22.45 ms

Table 1: computation times of the pure visual preemptive and hybrid RANSAC.

3.5 Conclusions & perspectives

We have presented an online, frame-to-frame, real-time hybrid motion estimation algorithm based on the preemptive RANSAC scheme. Thanks to a hybrid scoring of motion models, and a dynamic lagrangian computation, it is able to perform well in very difficult visual motion estimation cases, while relying on the inertial measurements only when strictly necessary. The fact that it is frame-to-frame avoids error accumulations and limits memory requirements. As next steps, we want to use a more elaborate distance measure between models, in order to have a better hybrid scoring. Another point of improvement could be to use more information to dynamically set the lagrangian factor, such as acceleration and linear translations measured by the sensors.

Chapitre 4: Localisation hybride

Below is a French summary of chapter 4: Hybrid Localization.

Ce chapitre se concentre sur les méthodes de « Simultaneous Localization And Mapping » (SLAM), qui consistent à estimer le mouvement de la plateforme tout en cartographiant l'environnement entourant celle-ci. Les données de camera et de capteurs inertiels sont fusionnées afin d'effectuer le SLAM pour l'appareil embarqué. Nous nous concentrerons sur l'estimation du mouvement de la plateforme.

Deux grandes catégories de méthode existent afin d'effectuer l'estimation du mouvement : le filtrage et l'optimisation. Après avoir étudié ces deux types de techniques, nous proposons une nouvelle méthode qui décompose le calcul de la pose en deux parties. Chaque partie fait appel à un type de technique. Il en résulte donc une méthode effectuant la fusion de données sur plusieurs niveaux.

Ce chapitre introduit les méthodes de SLAM de la littérature. Des expériences sur chacun des types de méthodes sont menées, afin de tirer le maximum d'informations possibles sur leurs avantages et inconvénients. Nous présentons ensuite une nouvelle technique d'estimation du mouvement hybride. Enfin, un matériel basé sur de la technologie infra-rouge est utilisé pour calculer une vérité terrain, et comparer notre méthode et celles de l'état de l'art.

Chapter 4: Hybrid Localization

Simultaneous Localization And Mapping (SLAM) consists in estimating the 3D motion of a platform in an environment (also known as ego-motion) and map its surrounding scene at the same time. Cameras and inertial sensors can be fused in order to perform an accurate and robust estimation, using the characteristics of each sensor. This chapter focuses on the ego-motion estimation part of the SLAM, which consists in computing in real-time its orientation and position.

Two main application domains utilize hybrid visual inertial SLAM: augmented reality and robotic navigation. The aimed applications in this thesis are mainly indoor navigation and augmented reality for handheld devices such as smartphones and tablets. As with the previous work developed in this thesis, robustness to complex scenes and motions is the main focus, while keeping a real-time performance.

Section 4.1 presents an overview of the state of the art on visual-inertial SLAM. Firstly, pure visual methods are described, as they often constitute the base of a hybrid algorithm. Visual **odometry** is introduced, with the 2D correspondences and 2D to 3D correspondences. Visual SLAM procedures are then described. Secondly, a state of the art summary is given on visual-inertial SLAM techniques. Augmented reality and robotic navigation possess their own methods, which are detailed.

In section 4.2, experiments on state of the art visual methods are conducted. An initialization technique for the SLAM algorithm is needed, as we need to create a 3D map to start the algorithm. An existing algorithm to perform this is detailed, the 7-point algorithm. In general, two main methods exist to estimate the ego motion based on 2D to 3D correspondences: **optimization** and **filtering**. Both display their own advantages and drawbacks, which are shown and studied with various experiments and tests.

A novel algorithm to perform visual-inertial odometry is introduced in section 4.3. It makes use of both optimization and filtering methods to estimate the ego-motion of the device with a sequential method, splitting the computation of the ego-motion in two phases: orientation and position. Orientation is first estimated using an optimization technique, the hybrid RANSAC that removes outliers and displays high robustness. Position is computed with a filtering stage that possesses a specific parameterization, reactive to the visual tracking quality and semantic information.

Section 4.4 shows results and conclusions of the proposed approach. We test our algorithm and compare it with state of the art techniques. A setup using infrared markers and cameras is used to record very accurate data on the ego motion of the device and are used as ground truth. Finally, computational complexity is studied and conclusions are drawn.

4.1 3D motion estimation: SLAM / Visual odometry

SLAM can be separated in two fields: dense-SLAM, where a very high proportion of points in the scene are considered, and feature-based SLAM, which performs a much more sparse estimation of points in the environment. Dense (or semi-dense) approaches have become increasingly popular recently [Schöps et al. 2014] [Engel et al. 2013] [Wendel et al. 2012], allowing to reconstruct many points of the scene with a whole-image alignment. While showing good efficiency, they remain very demanding in terms of computational resource. On memory side, the dense representation of the map also limits the size of the environment we can reconstruct, which is a drawback for applications such as navigation. Therefore, we focus our overview on systems relying on features to perform the SLAM algorithm.

4.1.1 Purely visual system

Estimating the ego motion of a platform is a complex task, requiring accuracy and robustness. Many sensors can be used to fulfill this purpose, from proprioceptive ones that measures only platform-related quantities (inertial sensors, wheels speed, GPS…) to sensors that records data from the external environment, deducing the motion regarding the temporal changes that occur (monocular cameras, stereo vision, laser, radars…). Monocular cameras are a very popular type of sensor in ego motion computation for many reasons. They are cheap, do not present high constraints in terms of weight or power, and their recording can be utilized for multiple purposes, making it a very flexible sensor, used for many years [Longuet-Higgins 1981] [Harris & Pike 1988].

4.1.1.1 Relationship between positions: matrices definition

As seen in section 1.4.1 , a 3D point $p = (x, y, z, 1)^T$ is projected on an image I_t onto a 2D point $X_t = (u_t, w_t, 1)$ with the projective equation: $X_t = KP_t p$ where K is the calibration matrix of the camera, and P_t is the projection matrix that describes the pose of the camera when frame I_t has been recorded. However, from direct 2D motion vectors, a relationship between point's measurements on different frame can be established. The geometric relationship between two separate frames I_t and $I_{t'}$ recording the same point is called the epipolar constraint. The fundamental matrix expresses this constraint:

$$X_{t'} F X_t = 0 \qquad (39)$$

The fundamental matrix is a 3x3 rank 2 homogenous matrix that encapsulates the geometric transformation between two recorded images. It is also known as the bifocal tensor of a two views system. To directly relate this matrix with the relative rotation and translation between the two cameras positions where the images have been recorded, one must use the essential matrix E. It can be computed based on the fundamental matrix:

$$E = K^T F K \quad (40)$$

Please note that the constraint is expressed in (39), but the points X_t have to be un-calibrated beforehand. The essential matrix is directly related to the relative rotation and position changes:

$$E \cong \hat{t}R \quad (41)$$

where R is the rotation matrix between the two views and $\hat{t}$ the skew-symmetric matrix based on translation vector t. This relationship is only correct up to a scale, as stated with the symbol $\cong$. These matrices F and E allow us to compute the 3D relative motion between two views only from 2D to 2D correspondences. This is a very valuable asset, leading to possible 3D interpretations only from the 2D camera recordings, without any preliminary knowledge on the scene.

4.1.1.2 Visual odometry

Performing this procedure without computing a global consistent map is called visual odometry. It aims at incrementally finding the trajectory and orientation of the mobile, possibly by building a local map of the environment, but without global optimization nor loop closure techniques. For monocular visual odometry, two categories of input can be utilized: 2D correspondences, or 3D to 2D correspondences.

- **2D correspondences methods**

Methods based on 2D correspondences consist in computing the essential matrix based on frame-to-frame motion vectors. We note the input motion vectors $v(i) = (X_i^{prev}, X_i^{curr})$. Each point's homogenous coordinates are noted $X_i^{prev} = (u_i, w_i, 1)^T$ and $X_i^{curr} = (u'_i, w'_i, 1)^T$. As defined in (39), each correspondence provides a constraint $X_i^{curr} F X_i^{prev} = 0$. If we put the values of F in a 9x1 vector $\tilde{F}$, the constraints take the form:

$$(u_i u'_i \;\; w_i u'_i \;\; u'_i \;\; u_i w'_i \;\; w_i w'_i \;\; w'_i \;\; u_i \;\; w_i \;\; 1)\tilde{F} = 0 \quad (42)$$

By stacking several lines of this constraint, one ends up with a system of the type $A\tilde{F} = 0$, which can be solved with classical techniques such as Singular Value Decomposition (SVD). Due to being a "calibrated" relationship between the views, only one property is available for the fundamental matrix: $\det(F) = 0$. This leads to the fact that at least seven motion vectors have to be provided to be able to compute F, but if eight or more points are used, the procedure is called the eight-point algorithm [Hartley & Zisserman 2003] [Longuet-Higgins 1981] [Hartley 1997].

If un-calibrated points are provided, then the essential matrix can be directly computed using another constraint: $2EE^T - tr(EE^T)E = 0$. This leads to the ability to compute the essential matrix from five points only, with efficient manners to compute it [Nistér 2004] [Hartley 2006].

As the minimal set to compute the essential matrix is not too prohibitive, RANSAC approaches have been utilized in combination with five points solvers [Nistér et al. 2006] [Lhuillier 2005] [Nistér 2005] [Mouragnon et al. 2006].

The operation of finding a 3D point coordinates based on the 2D projections and cameras poses is called triangulation. Further details on triangulation can be found in [Hartley & Zisserman 2003]. With the essential matrix computed, one can retrieve R and t from it with decomposition techniques [Nistér 2004]. Note that this decomposition leads to four possible solutions. The selection among those possible transformations is made by finding the decomposition that projects most 3D points in front of both cameras. When the 3D points are triangulated, we compute their depth for both cameras with every possible solutions for R and t, and keep the decomposition that projects the most points with positive depths.

The main advantage of 2D correspondences methods is that they only require knowledge of the camera calibration, but no information on the scene. The drawback is that the essential matrix is only defined up to a scale, leading to a complex relative scale computation that needs to be performed at every frame, based on the ratio between distances triangulated with the pose between the two views. Further details on this can be found in [Scaramuzza & Fraundorfer 2012].

- **3D to 2D correspondences methods**

In this case, a sparse 3D map of the scene is supposed to be computed, with appearance of each point recorded in the form of a descriptor (either a pixel patch or a higher level descriptor, as in 2.1.2.2). The input correspondences are therefore of the form (X_i^{curr}, p_i) where X_i^{curr} is the 2D projection of the 3D point p_i in the current frame. This procedure is known as the Perspective n Points (PnP) problem, n being the number of points used. On can employ a method similar to the 2D correspondences techniques, but this time to retrieve the projection matrix $P = [R|t]$:

- Every correspondence creates a constraint from the equation $X_i^{curr} = KPp_i$
- Stack the constraints to obtain a system of the type $A\tilde{P} = 0$, with $\tilde{P}$ being a vector form the projection matrix.
- Solve the system with a decomposition technique such as SVD.

This technique is known as the Direct Linear Transform (DLT) [Hartley & Zisserman 2003]. As each point creates two constraints, six correspondences or more are needed to compute the projection matrix. However, as seen previously, procedures such as RANSAC can be employed with a minimal solver. The rotation matrix only has three degrees of freedom, therefore retrieving the pose of the camera is a six DOF problem, and every point correspondences provide two constraints. Therefore P3P ($n = 3$) techniques have been highly popular in the literature [Haralick et al. 1994] [Nistér & Stewénius 2006] [Kneip et al. 2011]. For other solvers than minimal ones, the main difficulty lies in the complexity needed to solve the problem, with

relation to the amount of points provided. Some techniques display $O(n)$ complexity: [Moreno-Noguer et al. 2007] [Lepetit et al. 2008].

To conclude, an efficient way to perform visual odometry is to start with 2D correspondences techniques as they do not require any knowledge on the scene. Then, when a first local map is available with triangulation, a switch to a 3D to 2D correspondences can be performed. It is in effect more robust and accurate, as the geometry of the scene is available.

4.1.1.3 SLAM

The additional process in SLAM compared to visual odometry is to obtain a globally consistent map. This induces that every 3D point estimated position should be updated when observed by the camera, and loop closure needs to be performed. Two main strategies exist for the visual SLAM: **filtering techniques** (probabilistic) and **optimization-based procedures** (determinist).

- **Filtering techniques**

The SLAM problem can be treated as a probability density estimation problem. In this approach, the first two statistical moments (mean value and covariance) of the camera pose and 3D points are jointly estimated in a Bayesian filter, which is often either a Kalman filter (KF) or a **particle filter**. For the KF, an extended or unscented version is often used due to non-linearity of the system. KF-based techniques are described in Appendix A:. Particle filters are described in Appendix C:.

A general scheme of these filtering methods is the following. A dynamical temporal model of the system is used in the **state** of the filter, which are the physical values to be estimated: 3D orientation and position, along with their first derivatives, rotation speed and translation speed. A representation of the locations of the points is also included in the state. In the case of KFs, the covariance of both pose and points locations is directly accessible through a **covariance matrix**. For particle filters it is rather implicitly estimated by the particles distributions.

In each case, the procedure is separated in two steps: **propagation**, or time update, and **correction**, or measurement update. The propagation step consists in applying the physical model temporal behavior on the state (and covariance for KFs) of the filter. For instance, the estimated 3D orientation of the filter will be updated with the speed orientation of the state. Correction uses external measurement to correct the state with inputs from the outside world. In the monocular SLAM case, those are often motion vectors obtained from 3D to 2D correspondences.

This type of method was initially proposed by [Davison 2003] with an EKF system based on high quality points tracking, in this case SIFT features. The system has been further improved with the MonoSLAM algorithm [Davison et al. 2007]. This system is illustrated in Figure 4-1,

where the current camera view is displayed along with the 3D points map. Ellipses represent the uncertainty estimates by the filter, the bigger the ellipse is, the higher the covariance. As the EKF is a very generic filtering technique, specific parameterization can be used to further enhance its performance for the SLAM algorithm. In the case of MonoSLAM, an inverse depth parameterization is applied [Civera et al. 2008], providing a more stable system regarding poorly initialized points mean and covariance values. Numerous extensions and studies on the EKF-SLAM were proposed in the literature [Paz et al. 2007] [Bailey et al. 2006] [Solà et al. 2011]. The EKF probabilistic information can also be used in a RANSAC strategy to generate hypothesis based on the probabilistic distribution estimated on the filter [Civera et al. 2010].

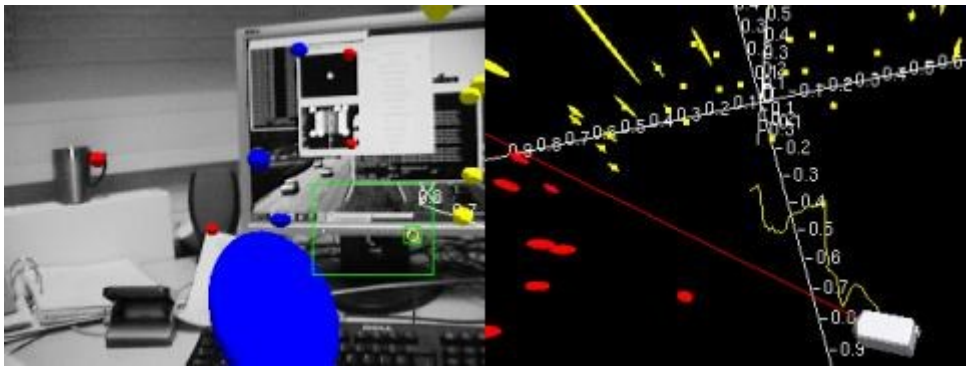


Figure 4-1: Illustration of the MonoSLAM technique [Davison et al. 2007]**. Image courtesy of [10]. On the left, the current frame with 3D points projected in the 2D camera view. On the right, the 3D map, as well as camera pose and trajectory, are shown.**

Many more filtering techniques other than the EKF can be used to solve the monocular SLAM problem. [Holmes et al. 2009] show an Unscented Kalman filter (UKF) technique, proposing a Square Root adaptation to cope with computational cost considerations. The fastSLAM [Montemerlo 2003] technique was introduced mainly for robots, using a particle filter scheme to estimate the pose of the platform, and each particle estimated its own map. An adaptation of this technique for handheld cameras more prone to augmented reality is introduced in [Eade & Drummond 2006]. Further extension of this strategy is described in [Lee et al. 2011], where the particles are drawn from a 5-point RANSAC technique.

- **Optimization based strategies**

Another possible technique to solve the Localization problem of the SLAM is to use optimization techniques. M-estimators (3.1.1.2) are the gold standard way to perform this. The **Parallel Tracking And Mapping** (PTAM) algorithm separates the Localization and mapping procedures in distinct threads, to make use of the multiple cores present on modern computers and smartphones architectures [Klein & Murray 2007]. The Localization is performed in real-time using an M-estimator, while the mapping is done on a thread by applying **Bundle Adjustment** (BA).

[10] http://www.robots.ox.ac.uk/ActiveVision/Research/Projects/2003ajd_monoslam/project.html

The BA technique consists in rectifying the previous camera poses and 3D point's locations. The energy that is minimized is based on the reprojection error of every point on the frames considered, with a Levenberg-Marquardt iterative approach [Marquardt 1963]. This type of technique possesses the advantage to reconsider the map as a complete system to be solved, which allows efficient loop closure. Further details on BA can be found in [Triggs et al. 2000].

This mapping thread using BA is not constrained with real-time requirement, and is only performed on a few key frames that are selected using criteria such as sufficient performance of the tracking. This task split is a great asset for applications such as augmented reality that needs many features to be tracked in the map. Figure 4-2 is an example of the PTAM applied to augmented reality, where a dominant plane is extracted from the 3D point's locations to perform various tasks.

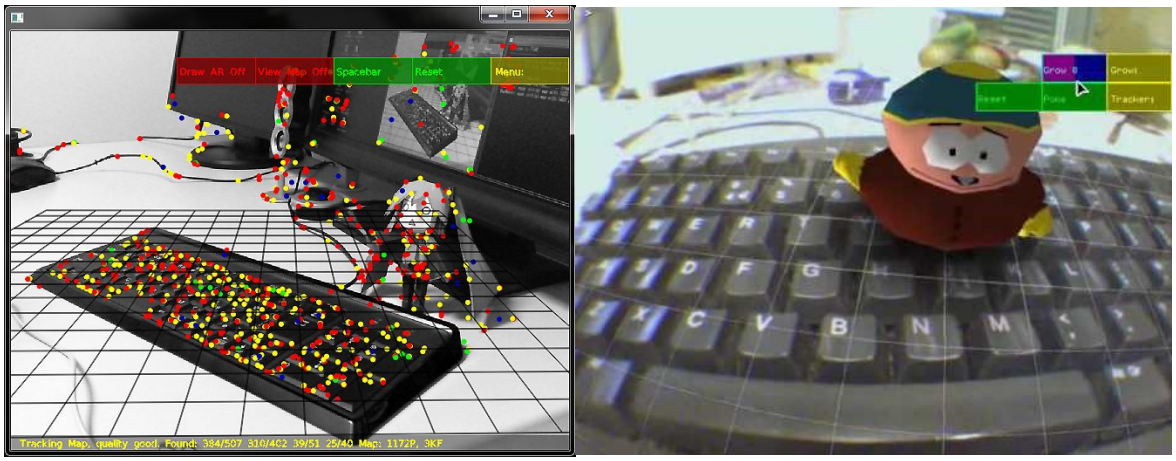


Figure 4-2: An example of PTAM applied for augmented reality purposes. On the left, points of the map are displayed along a grid that represent the dominant plane. One the right, the plane is used to display objects in the scene. Image extracted from [11]

Indeed, the real-time localization tasks only consider points observable by the pose of the camera, while the mapping is not strictly constrained in computation. The PTAM system have been extended to handheld devices [Klein & Murray 2009], including rolling shutter considerations and a two-staged tracking process using coarse-to-fine pose computation.

More recent works treat of additional robustness strategies, such as fast re-localization in case of poor tracking performance in the localization thread [Martin et al. 2014]. Other techniques make use of optimization and BA to compute the pose of the camera. [Mouragnon et al. 2006] use a local BA to perform SLAM on ground vehicles, in which a global BA is not needed as the scene often changes due to the high speed of motion.

A comparison of filtering and optimization + BA systems was made in [Strasdat et al. 2010]. The authors came to the conclusion that, while filtering is still a viable option for very limited

[11] http://www.robots.ox.ac.uk/~gk/PTAM/

resource platforms, optimization + BA systems display the highest accuracy per amount of computational operation performed.

4.1.2 Hybrid SLAM

Inertial sensors coupled to monocular cameras are present on many systems. Therefore hybrid SLAM procedures using these two sensors have been very popular recently in the literature. Two types of applications are aimed: augmented reality and mobile robotics.

4.1.2.1 Fusion for augmented reality

A first way to use inertial sensors in visual augmented reality is simply to incorporate the accelerometer as an indicator of the gravity. This allows feature descriptors to be oriented along this axis, making them robust to in-plane rotations [Kurz & Ben Himane 2011]. This technique is especially valuable for augmented reality used on vertical planes, as the in-plane rotation invariance is much easier and cheaper to guarantee [Kurz & Benhimane 2012]. Gravity-based orientation can also improve localization on a higher scale, for instance to retrieve image's locations in a wide environment [Arth et al. 2012].

Augmented reality techniques can also be improved in terms of robustness to quick rotations of the platform using gyroscopes, which measure the orientation changes. In [Klein & Drummond 2004] and [Klein & Drummond 2003], a visual tracking system is improved with tightly integrated sensor fusion. The authors use gyroscopes to predict the motion to be estimated with visual techniques, as well as the amount of blur that will occur. Similar methods are based on a gyroscope coupled to a magnetometer, where the orientation estimation given by the inertial sensors is applied to predict to 2D locations of the points from the 3D map [You, U. Neumann, et al. 1999] [You, Ulrich Neumann, et al. 1999], leading to an increase in terms of robustness to high motions and blur.

Early work demonstrated by [Azuma et al. 1998] underlined the need of hybrid strategies to perform outdoor augmented reality techniques, as the environment is less constrained. [You & Neumann 2001] included the measurement of a gyroscope in an EKF, performing two different measurement updates, corresponding to each type of sensor: one for the visual measurements, one for the inertial measurements. The different frequencies of the sensors can lead to a different integration of these in the localization procedure. [Lang et al. 2002] propose an inertial based estimation, where the visual measurements are used as a correction, as the camera display a lower frequency. Integration of inertial measurement in tracking system of a 3D model of the scene is studied in [Bleser & Stricker 2008], including a comparison of several types of fusion.

An addition of inertial sensors to the PTAM algorithm is presented in [Porzi et al. 2012] for Android© smartphones. The localization thread is simplified to consume less computational resources. Inertial measurements are integrated at every time that they are recorded, to predict

the pose of the next frame. In the complete system, this allows not to use a coarse-to-fine approach to localization, but rather an inertial-based motion prediction and visual-based correction of the prediction with the M-estimator.

Other techniques are especially designed for indoor applications. Some constraints on the scene can be applied thanks to the particular geometry inside a building. The Manhattan world hypothesis consists in considering that the scene is composed with planes that are either perpendicular or parallel with respect to each other. Applications for both monocular [Coughlan & Yuille 1999] and stereovision [Furukawa et al. 2009] motion estimation methods were designed based on this principle.

4.1.2.2 Fusion for mobile robotics

In mobile robotics, the problematic is not the same as augmented reality when it comes to performing inertial visual fusion. In a general manner, an inertial navigation is used as the basis of the approach, while visual measurements prevent the drifts to influence the estimation. Systems such as the one presented in [Roumeliotis et al. 2002] display this type of fusion, here adapted to spacecraft navigation. [Diel et al. 2005] introduces epipolar constraints as a valuable addition to inertial navigation based on KFs. [Lobo & Dias 2003] use a combination of the vertical reference measured by inertial sensors and vanishing point computed in the frame to retrieve the focal length of the camera and reconstructing the scene. More recent works apply other constraints coming from the vision sensors, such as ground planes [Panahandeh et al. 2012].

Adaptation to the classical EKF technique has been highly popular in the navigation applications. In a similar manner as [You & Neumann 2001] for the augmented reality strategies, [Strelow & Singh 2003] apply an Iterated Extended Kalman Filter (IEKF), where every measurement is treated in the filter at the time of its recording.

[Mourikis & Roumeliotis 2007] introduce a Multi-State Constraint Kalman filter (MSCKF) with a motion model constrained with every observed point separately. Rather than augmenting the state with every point, the state includes a certain amount of previous poses. It should be noted that this procedure is quite demanding in computational resources, which limits it to offline processing. This technique is improved in [Li & Mourikis 2013], leading to a performance improvement that is reported to be better than the conventional EKF-SLAM technique, without storing the mapped points. Another adaptation of the technique is introduced in [Li et al. 2013], with a modified computation of the covariance matrices as well as the consideration of rolling shutter distortions.

The EKF and its variants are not the only options when it comes to inertial visual navigation. [Yap et al. 2011] present a particle filter applied to navigation. [Durrie et al. 2009] also makes use of particle filtering to perform localization of unmanned vehicles. Other navigation techniques apply more biologically inspired strategies, such as the RatSLAM [Milford et al.

2004] that incorporates a pose cells based estimation to cope with ambiguous landmarks measurements.

4.2 Hands-on experimentation and discussion

To further study the advantages and drawbacks of 3D motion estimation techniques, we implemented several procedures in order to perform tests. This has led to a better understanding of the advantages, drawbacks, requirements, and parameterization of these techniques. Once these experiments conducted, we discuss on the design of a novel localization technique.

4.2.1 Initialization method

As stated previously, the SLAM algorithm requires a 3D "base" map to start working. To perform this initialization, a few methods are possible:

a) To have a 3D model of the scene that we try to map on the observed frames.
b) To recognize a planar object in the scene (chessboard for instance), which can be used as a starting point to compute motion and map other objects, using planar homography decomposition techniques [Faugeras & Lustman 1988].
c) To compute a map based on 2D to 2D correspondences odometry, such as a fundamental (or essential) matrix computation strategy [Hartley 1997] [Nistér 2004].

The method a) is the easiest to use, as we can start the SLAM procedure right away, but needs strong knowledge on the scene, and therefore is not applicable in a general context. The technique b) is more general, but still poses a heavy requirement on the scene: the presence on the scene of a known object. The procedure c) is the most complex, but also the most general, as no requirement on the scene is needed. Thus, we decided to apply method c). in the context of embedded platforms, where the aim is to perform video applications on many different scenes. The initialization procedure simply consists in performing a translational motion of approximately 15 centimeters. The motion should be translational because only this type of motion allows to retrieve some information on the depth of the objects in the scene.

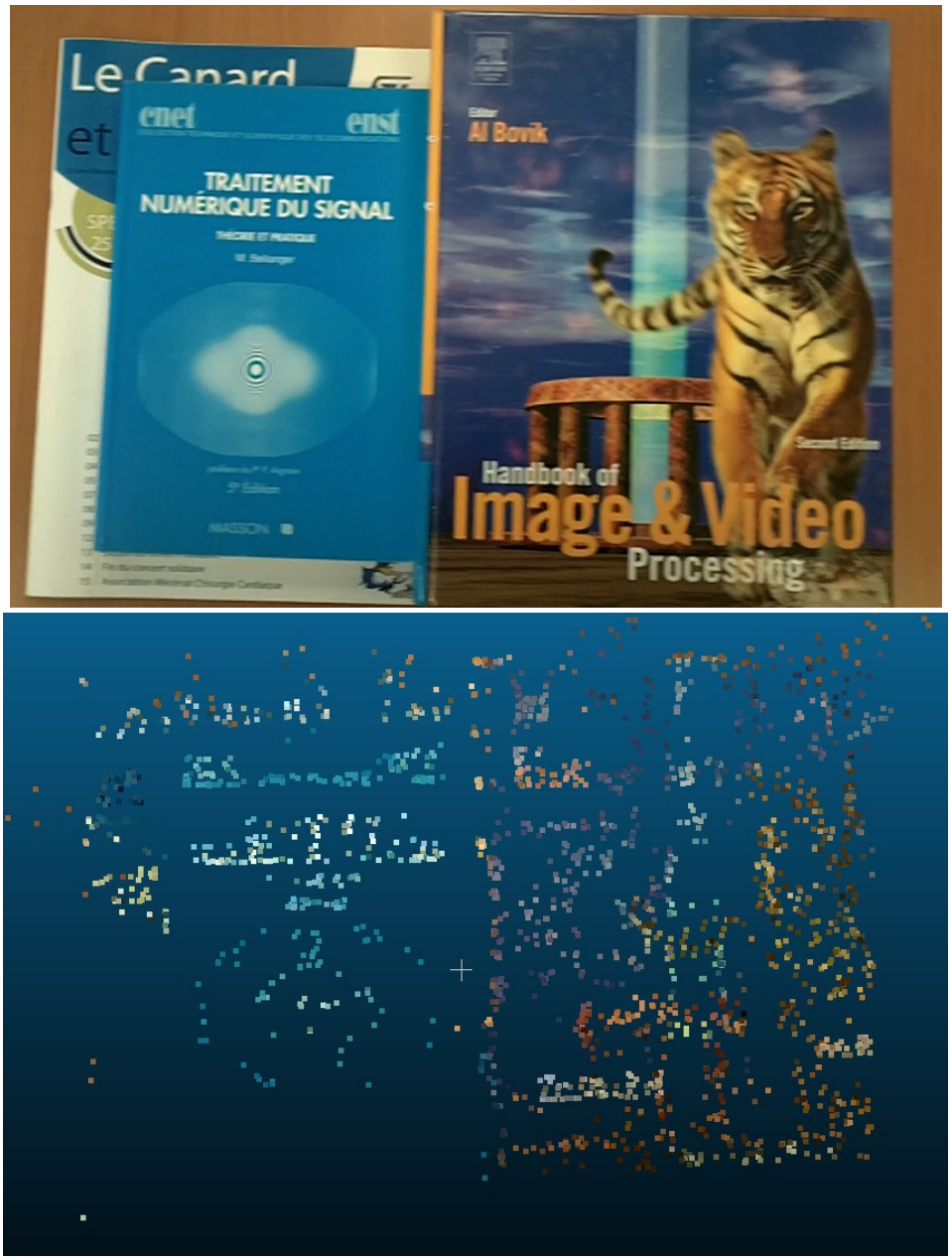
As presented in 4.1.1.2 , 2D correspondences can be used to estimate the fundamental (or essential if the points are un-calibrated) matrix. Two main strategies exist: using every correspondence in a Direct Linear Transform (DLT) system, or perform the estimation in a RANSAC procedure 3.2.1 . As one focus of the thesis is to use inexpensive motion estimation, which may generate many outliers, we chose to implement a RANSAC algorithm procedure for our system. The inputs are motion vectors generated by FAST/BRIEF (or IBRIEF) matching.

Two main techniques exist to compute the relationship between the two views: compute the fundamental matrix based on a 8-points (or 7-points) procedure, or un-calibrate the points to then use a 5-points algorithm. As the usage of a 5-point algorithm requires accurate calibration,

we chose to rather use a 7-point algorithm, as we want the technique to be applicable on many platforms without heavy requirements in terms of calibration.

The 7-point algorithm is very similar to the 8-point algorithm, but requires one less motion vector to produce a solution, which decreases the necessary amount of iterations to be performed in the RANSAC algorithm. Another advantage of the 7-point algorithm is that it only generates from one to three solutions for the fundamental matrix, opposed to up to ten solutions for a 5-point algorithm. Further details on this subject can be found in [Hartley 1997], [Hartley & Zisserman 2003], and [Nistér 2004].

It was tested on sixteen scenes to assess the performance of this approach, and have a better understanding of its limitations. Figure 4-3 illustrates a scene with two views of the triangulated point cloud using the transformation computed by the 7-point RANSAC algorithm.



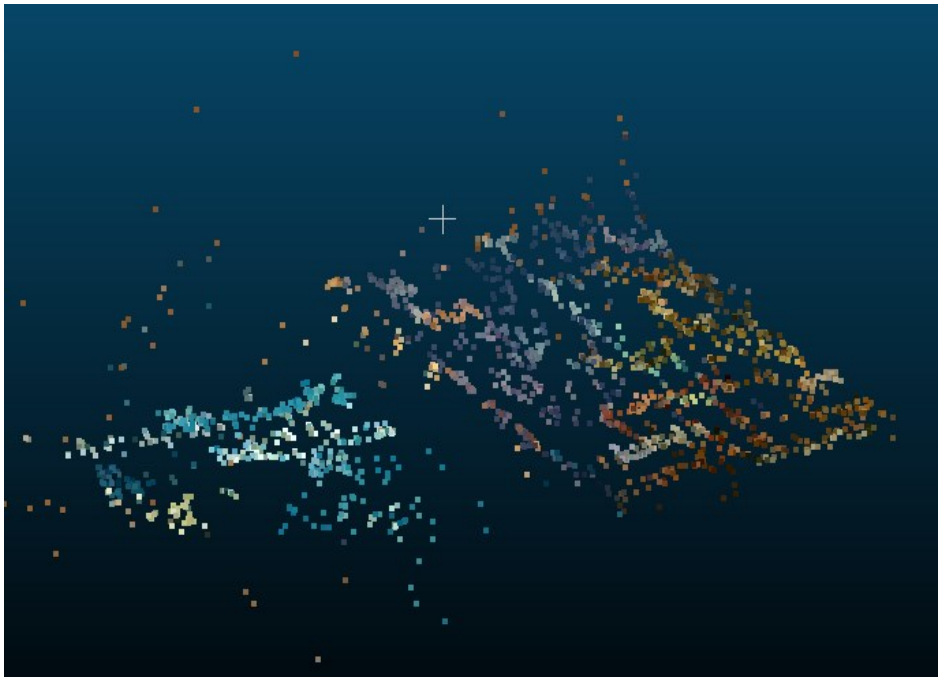


Figure 4-3: An example of a scene (top) and its triangulated 3D point map (middle and bottom, with different viewpoints) using a 7-point algorithm in a RANSAC procedure.

The main characteristics we were able to underline are that:

- If no particular assumption on the scene has to be made, assumptions on the motion performed are important. A pure rotation does not give any information on the structure of the scene. Therefore a sufficient translation needs to be performed for the technique to perform efficiently.
- The technique is quite sensitive to the outlier percentage. If we have a limited amount of iterations (or models for the preemptive procedure), the technique does not perform satisfyingly on difficult scenes.
- No particular assumption on the scene is needed. Visually speaking, the algorithm performed well on diverse scenes, from desks to corridors, several depths to flat scenes (walls).

To conclude, the 7-point RANSAC algorithm is a solid base to generate a 3D map at the beginning of the sequence without previous information on the scene, which is then used to perform SLAM techniques.

4.2.2 Localization techniques based on 2D-to-3D matching

Having the 3D map initialized by a procedure as shown in 4.2.1 , monocular SLAM can be performed. As demonstrated in [Strasdat et al. 2010], the BA mapping techniques shows superior results, especially if the platform presents parallel computing capabilities that allows keyframe-based BA. A point that remains relatively open is the localization technique that should be employed. To further consider the characteristics of state of the art methods, it was decided to implement two typical ones and test them on relevant sequences. A succinct description of those two techniques will be given, along with the remarks made after the testing.

One procedure applies filtering to compute the pose (here an UKF), while the other one makes use of optimization.

- **Unscented Kalman based Localization**

The first method implemented is the UKF-SLAM. While the EKF-SLAM has been more popular in the literature, the UKF is more flexible in the sense that it does need the explicit representation of the Jacobian for each change in the motion model. The procedure used is the same as described in [Holmes et al. 2009]. The UKF technique is implemented in a square root version, by directly propagating a Cholesky decomposition of the covariance rather than the covariance itself. More details on the Unscented Kalman Filter are given in Appendix A.3 .

As the results of the UKF-SLAM and EKF-SLAM are similar in terms of accuracy [Crassidis & Markley 2003], this technique shall be considered as a generic filtering-based SLAM technique.

- **RANSAC with PnP**

The strategy used here is to perform a PnP algorithm (see section 4.1.1.2) in a RANSAC procedure. The PnP method selected was the EPnP one [Lepetit et al. 2008]. The goal here was to test the potential of optimization based approaches. Rather than using an M-estimator for this test, it was decided to rather apply a RANSAC procedure. In effect, as the BRIEF descriptors matching produces the motion vectors, many outliers may emerge. Thus, an iterative estimator may converge on a wrong model, while a "memory-less" one is much more robust to this type of high outlier presence.

Note that the implemented RANSAC is a preemptive one (see 3.2.3). This induces that we use a limited number of motion models. We decided here to utilize five hundred models, which seemed like a realistic value with regard to our experiments on the Hybrid RANSAC.

- **Results on tested sequences**

We tested the approaches on a few sequences, which helped figuring exactly the advantages and drawbacks in our context. We started the sequence by an initialization via the 7-point RANSAC that was the same for both methods of localization. Figure 4-4 displays a 3D map computed with the 7-point algorithm on the video frame.

Figure 4-4: A 3D map computed with the 7-point RANSAC algorithm. The smaller the circle is displayed, the closest is the point.

The evaluation of both methods was then performed by looking frame-by-frame that the 3D model was well superimposed on the scene, as the aim of this process is only to have a qualitative estimation of the performance of the two techniques rather than an accurate quantitative comparison.

In terms of performance with regard to simple scenes with few outliers, both methods responded satisfyingly. The 3D model was well projected for the two techniques, even if, for the UKF-SLAM, the dynamic models can create some latency in the reaction when a fast change of direction occurs, as demonstrated on Figure 4-5. The top frame shows a regular frame with the model well-mapped on it. The bottom frame occurs right after a change in direction from a left translation to a right translation. This leads to a slight offset in the localization that can be seen: the points on the bottle are not well-aligned with it, especially points that are close to the camera (see the ones inside the red circle).

Figure 4-5: A change of direction occurs, leading to the UKF dynamic model prediction to offset a bit the localization. Top displays a frame in the sequence where the model is well tracked, bottom shows that right when the changes of direction happens, the model is not well mapped on the scene.

The behavior of the two procedures in presence of a heavy amount of outliers is very different. Filtering is based on the physical model of the UKF-SLAM, which leads to the pose drifting away from the correct one in case of high amount of outliers. While not ideal, this behavior leads to a temporal consistency of the pose. For the EPnP RANSAC technique, if the amount of iterations was sufficient and the algorithm has found a correct model, the estimation is satisfying. Meanwhile, if the best model found is not created from inliers only (because not enough iterations were used), the procedure gives a poor result.

Figure 4-6: Two frames from the same video as Figure 4-4. Top shows the first frame, bottom row the second. Left frames show the UKF-SLAM results, right frames show the PnP RANSAC results.

Figure 4-6 is an example of the behavior of the procedures. The map computed is the one presented in Figure 4-4. We remind that the correspondences are computed between the first frame (Figure 4-4) and the current frame. Two frames that are very close are extracted (one is on top row, the other on the bottom row). The aim of this test is to validate the reliability of the techniques with very similar inputs.

On top row, a first frame where the UKF-SLAM (top left) estimates a pose that is quite far from the correct one. The PnP RANSAC (top right) is much more accurate, the pose estimated seems correct, as the map superimposed corresponds to the one computed at the beginning on Figure 4-4. This shows that on this frame, with the same motion vectors, the UKF-SLAM underperforms compared to PnP RANSAC. On the other hand, the bottom row shows a frame where the UKF-SLAM (bottom left) behaves similarly as previous frame. The PnP-RANSAC (bottom right) however, performs poorly and computes a pose that seems to be far from the correct one.

The main conclusion that can be drawn for this is that the UKF-SLAM has a much more coherent behavior with respect to a scene with a respectable proportion of outliers. PnP RANSAC displays a much more binary behavior, either finding the correct pose even with outliers, or failing to retrieve it, depending on its ability to generate a good model in the RANSAC procedure. In this approach, very similar inputs can lead to very different results.

4.2.3 Discussion on the state of the art

Having performed an overview and some tests on the state of the art methods for the SLAM, we are now able to draw some requirements and characteristics on the localization technique to be designed. As Bundle Adjustment has proven to be the overall best solution to mapping in this context, our interest was to focus on improving the localization part of SLAM. We highlight here the well-known flaws and issues of classical visual-inertial odometry algorithms, especially on handled devices.

First, the lack of texture in some images impacts heavily the visual measurements, making classical filtering or optimization unstable. It appears that a direct integration of visual motion vectors in a filter without discarding outliers can severely degrade the ego-motion estimation, even if inertial readings are also fed to the filter. Optimization techniques can deal better with outliers, but can also severely diverge in case of a very large proportion of incorrect motion vectors.

Quick wrist motions create large rotations, leading to blurry content and important changes of viewpoint. Current techniques rarely manage to keep satisfying ego-motion estimation under such conditions. Moreover, the embedded inertial sensors that can be found in smartphones and tablets can be very noisy. While inertial readings provide a more or less reliable 3D rotation estimation, computing position based on their measurements is a very inaccurate process, even with precise calibration: the only way to compute position is by a double integration of the accelerometer, which leads to heavy drift.

A great advantage of filtering methods is the dynamical physical model that provides temporal consistency of the pose estimation. While some optimization methods also make use of a temporal propagation [Klein & Murray 2007], some kind of temporal constraints in the motion provides more robustness to very complex estimation cases.

In a major part of state of the art techniques, the inertial visual fusion is only performed in one stage of the algorithm. For embedded platforms, inertial sensors are used: either to propagate the state of a filtering technique [Aksoy & Alatan 2014], to provide a first estimation of the rotation [Klein & Drummond 2003] are integrated as some complementary measurements to visual sensors [Li & Mourikis 2012], or to give orientation information on the feature points in the scene [Kurz & Benhimane 2011]. We feel that the fusion could take place in multiple stage of the localization technique, in order to extract the highest possible amount of information from their measurements.

Lastly, inertial sensors and cameras can provide more information than their sole measurements in the context of this thesis. For instance, inertial sensors can be used to determine if the wearer is walking or not [Feliz et al. 2009], or vision used to recognize the type of the scene that is observed [Filliat 2007]. Therefore we could also integrate this type of semantic information in the localization process.

4.3 Multiple Level Fusion Odometry

We present a novel approach to the localization part, or odometry, in the hybrid visual-inertial SLAM. The combination of inertial measurements together with computer vision techniques is performed in a **Multiple Level Fusion Odometry** (MLFO). To combine the best characteristics of filtering and optimization approaches, the two are integrated in our approach. Figure 4-7 displays the global algorithm scheme.

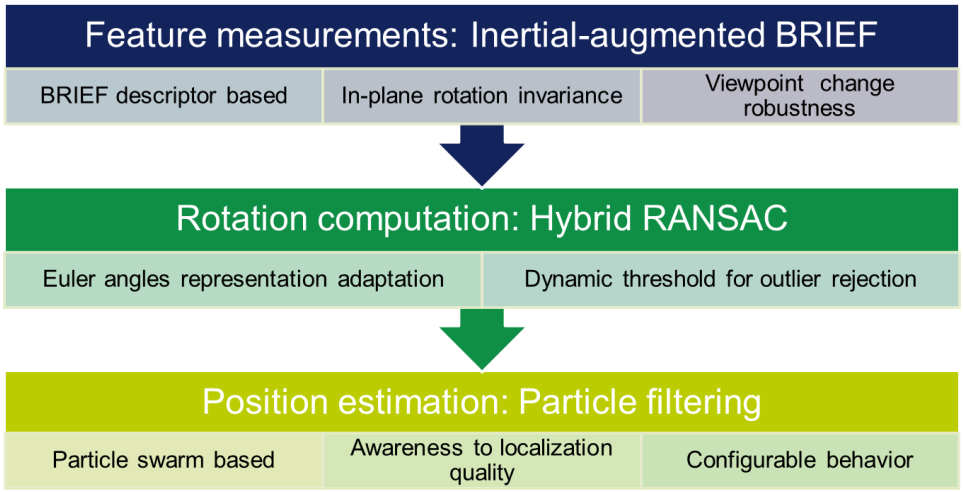


Figure 4-7: The global localization algorithm steps.

As seen previously for the Hybrid RANSAC experiments, MEMS inertial sensors offer an accurate and robust estimation of the orientation of the device. However, for position, the measurement are much less reliable, as only the accelerometer gives an information on it, and the double integration needed leads to noise and drift. To better take into account these characteristics, we split the ego-motion estimation in two, and perform several types of fusion for each step, including feature measurements.

Firstly, visual features are described with IBRIEF (see 2.3) descriptors, leading to less errors under rotations in the feature matching process. A separation of the 3D rotation estimation of the camera and the calculus of its position in the 3D space is made, in order to provide the most optimal use of the combined strength of vision and inertial sensing.

Secondly, the 3D camera rotation is computed with a hybrid RANSAC-based algorithm (3.3). This method allows us to cope with high rotations and complex scenes, such as texture-less sequences. It also removes the outliers, which is important to perform efficient localization.

Finally, with 3D rotation known, a visual measurement-based particle filtering is performed to compute the camera position. Embedded inertial sensors are very noisy, therefore it is very challenging to have a valid position estimation from their measurements. Rather than using

directly the inertial measurement, we guide the particle spreading with a pedestrian step detection technique derived from accelerometers data.

4.3.1 Estimating Rotation with Hybrid RANSAC

Adapting the hybrid RANSAC directly to localization would require computing a motion model that includes rotation and position from inertial measurements. But as position is computed from a double integration of the accelerometer, which is noisy, the position deduced from this method cannot be utilized reliably. Thus we decided to use this method to compute ego-rotation only.

4.3.1.1 Adapting the algorithm to estimate 3D rotation

The inertial data are modeled as Euler angles $Inert(t) = \alpha_t, \beta_t, \gamma_t$ as the relative yaw, pitch and roll differences that occurred between frame $t-1$ and frame t. The visual measurement are the 3D / 2D correspondences between the 3D map modeled in the SLAM and the current 2D frame with its extracted features. The motion models $h(j)$ could be taken as rotation matrices, but it would lead to a high complexity both in model generation and in error computation $\epsilon(h(j), v(i))$, since we would need to project exactly every point and then compute its reprojection error.

To avoid these costs, we decided to model the rotation representations as Euler angles $h(j) = (\omega_x, \omega_y, \omega_z)$. This will lead to approximations that are acceptable when dealing with small angles. Measurements are first taken as couples (X_k, p_i), with $X_k = (x_k, y_k, z_k, w_k)^T$ a 3D point in homogenous coordinates, and $p_i = (u_i, v_i, w_i)$ a 2D point of the current frame. Those measurements are converted to motion vectors, to make the motion model generation less expensive.

K is the 3x3 intrinsic parameters matrix of the camera. P is the estimated 3x4 projection matrix from the previous frame. For each measurement, the 3D point is projected into the previously estimated camera pose according to:

$$p_k^{proj} = K * P * X_k \tag{43}$$

Measurements $v(i)$ are now couples (p_k^{proj}, p_i). We note du_i the horizontal component of the motion vector:

$$du_i = \frac{u_k^{proj}}{w_k^{proj}} - \frac{u_i}{w_i} \tag{44}$$

With similar computations, dv_i is calculated as the vertical component of the motion vector. We now need to generate the motion models $h(j)$, with a minimal number of visual measurements $v(i)$. While an exact computation of the rotation would require fairly complex methods, an approximation of it can be performed rapidly with two measurements $v(i)$ and $v(i')$, as one is not enough to compute the three rotations. The average motion is interpreted as the rotation in both directions. We compute yaw (with a similar technique for pitch ω_y) as:

$$\omega_x = (du_i + du_{i'})/2\rho_x \tag{45}$$

Where ρ_x, ρ_y are the respective scaling factors to convert yaw into horizontal translation and pitch into vertical translation in the 2D frame. Computation details can be found in [You, U. Neumann, et al. 1999], and are solely based on focal length of the camera and resolution of the image. Roll is calculated with a difference of angles in polar coordinates:

$$\omega_z = atan2(dv_{i'}, du_{i'}) - atan2(dv_i, du_i) \tag{46}$$

In a preemptive RANSAC procedure, considering an inlier rate τ_{inlier}, the probability to generate a model withtout outlier is $(\tau_{inlier})^m$, m being the number of measurement used to create a model. In our approach, $m = 2$, inducing a high probability to generate an outlier-free model. This is highly beneficial compared to other rotational models that need a much higher m. $\delta\big(h(j), Inert(t)\big)$ is simply the L2-norm between the two Euler angle representations. The error function $\epsilon(h(j), v(i))$ is chosen as the reprojection error.

4.3.1.2 Adaptive threshold to point distance

A novel variable constraint with respect to the depth of the 3D point observed $Dep(X_p)$ is added in the algorithm. Indeed, the closer a point, the more sensitive it is to translation motions that are here considered as noise, as we only want to estimate rotation. Therefore we have a variable threshold T_i, for each measurement $v(i)$:

$$T_i = T + \frac{\hat{V}_{t-1} * Dep(X_p)}{f} \tag{47}$$

Where $\hat{V}_{t-1}$ is the predicted velocity by the particle filter (to be described in section 4.3.2), and f is the focal length of the camera. For the best model $h(j_{best}) = (\omega_x^{best}, \omega_y^{best}, \omega_z^{best})$, any point $v(i)$ with $\epsilon(h(j_{best}), v(i)) > T_i$ is considered as an outlier. This allows the approach to better estimate the rotation without undergoing the perturbations due to translational motions, without adding too many computations.

4.3.2 Particle Swarm strategy to compute position

As stated previously, inertial sensors offer a good complementary estimation to vision when it comes to rotational motion. For camera position estimation however, they do not provide reliable information. On the other hand, visual based motion estimation also has difficulties regarding repetitive or low textured contents, typically found for example in large rooms. Therefore there is a need for a flexible position estimation, which could cope with very low quality measurements without drifting too much, and re-converge when meaningful information is provided back. Those considerations drove us to apply a particle swarm filter for the 3D position and velocity of the camera.

4.3.2.1 Particle filtering technique

Particle filtering is widely used in many domains of computer vision, from tracking to odometry. Also known as sequential Monte-Carlo estimation, it is a density estimator that utilizes a sampling strategy to represent the posterior density of the state probability [Lui & Chen 1998]. The approach proposed here is based on the particle swarm technique. Based on the 3D/2D visual measurements $v(i)$, now supposedly outlier free with the hybrid RANSAC procedure, an estimation of the camera position and velocity is performed. The estimated 3D position at frame t is noted $\widehat{D}_t$, the estimated velocity is noted $\hat{V}_t$. Each particle (which is a virtual camera) ξ_l^t has its own position d_l^t and velocity v_l^t. The algorithm consists in two steps. First, we perform propagation:

$$d_l^t = d_l^{t-1} + v_l^{t-1} \tag{48}$$

The probability of each particle with respect to the measurements $\pi(\xi_l|v(1,\dots,N))$ is then computed. In order to estimate these probabilities, we use the rotation calculated previously, $h(j_{best})$. We convert the Euler angles to an axis k_{best} and angle θ_{best} representation. Then the Rodrigues formula is used in order to obtain the corresponding rotation matrix:

$$R_{best} = I + K_{best} sin\theta_{best} + {K_{best}}^2(1 - cos\theta_{best}) \tag{49}$$

with K_{best} being the cross-product matrix of the vector k_{best}. A projection matrix P_l is therefore generated for every particle: $P_l = [R_{best}|d_l]$. Inlier measurements points are projected into the frame as shown in (43) for every particle, with projection matrices P_l. The reprojection error of measurement $v(i)$ for particle ξ_l^t is noted ε_i^l. $\pi(\xi_l^t|v(1,\dots,N))$ is computed as:

$$\pi\left(\xi_l^t\middle|v(1,\dots,N)\right) = 1/\sum_i \varepsilon_i^l \tag{50}$$

The estimated position and velocity are calculated as the probabilistic-weighted average of the particles.

$$\widehat{D}_t = \sum_l \frac{\pi\left(\xi_l^t\middle|v(1,\dots,N)\right)}{\pi_{tot}} d_l^t \tag{51}$$

with $\pi_{tot} = \sum_l \pi\left(\xi_l^t \middle| v(1,\dots,N)\right)$. The particle with highest probability, ξ_{best}^t, is now used to compute the velocity of each particle. A novel manner to adapt dynamically the filter to the quality of tracking is utilized in the velocity calculus:

$$v_l^t = v_l^{t-1} + \kappa(d_{best}^t - d_l^t) + (1-\kappa)r(l) \qquad (52)$$

$r(l)$ is a random 3D vector, which is generated according to a Gaussian distribution $\mathcal{N}(\mu_r, \sigma_r)$. The choices of (μ_r, σ_r) will be explicated bellow. κ is a factor that dictates how heavily we want to emphasize the convergence to the best particle found in the current step.

4.3.2.2 Specific parameterization of the filter

To make the filter reactive to the tracking quality, κ is set according to a relationship between the inlier rate found in the Hybrid RANSAC τ_{inlier}, the absolute reprojection error of the best particle ε^{best}, and a typical reprojection error we want to achieve ε^{typ}:

$$\kappa = \tau_{inlier} e^{-\left(\frac{\varepsilon^{best}}{\varepsilon^{typ}}\right)^2} \qquad (53)$$

Thanks to this specific parameterization, our particle filter reacts to the quality of the estimation, heavily diffusing the particles in case of very poor tracking, while providing a high convergence to the best particle in case of high quality tracking estimated.

A requirement stated while building this approach was to have an elegant manner to add complementary semantic information available. We wanted this to remain a simple process, to avoid overly complex parameterization. We choose to offer a simple configurability of the filter. The two parameters that can be tuned are (μ_r, σ_r), the characteristics of the Gaussian distribution for particle velocity calculation.

μ_r represents a typical velocity expected for the current frame. For instance, with the accelerometer, one can quite easily determine if the person holding the device is walking or not [Brajdic & Harle 2012]. Therefore, if the person is detected as walking, μ_r is calculated as:

$$\mu_r = R_{current} * \begin{pmatrix} 0 \\ 0 \\ S_t/fps \end{pmatrix} \qquad (54)$$

with $R_{current}$ being the current estimated rotation matrix, S_t the typical velocity of step and fps the frame rate of the camera. This parameterization will orientate the particle spreading forward, and help the algorithm to cope with walking motion, without explicitly including a

motion model in the approach. For the moment, σ_r is left constant, but this parameter will be tuned in future versions of the approach.

4.4 Results & conclusion

A comparison of the MLFO approach with other methods is presented here. To perform this, we used an Optitrack© based setup, which is composed by infrared markers and cameras to record an accurate pose estimation of the platform. This allows us to measure a ground truth (or at least highly accurate) error for every method. Thus, we are able to perform a quantitative and qualitative comparison of every tested strategy. We oppose our MLFO technique to state of the art methods, as well as dimensioning its computational time. Finally, conclusions are drawn.

4.4.1 Setup

Evaluating with accuracy odometry algorithms is a complex task, as it requires knowing the exact true motion of the platform. Therefore we decided to use the infrared tracking technology, which is widely used for motion capture in a lot of applications (movies, video games, virtual reality). A platform was built, containing the smartphone with infrared markers attached to it. Eight infrared cameras record their motion, while the smartphone records the video and inertial readings. The recording accuracy of a marker by infrared cameras is very high (~1mm accuracy), and is performed a high rate (250 fps). Thus it is considered as ground-truth in our experiments, because its accuracy is highly superior to the one that a monocular-inertial SLAM can provide. Figure 4-8 displays the setup used.

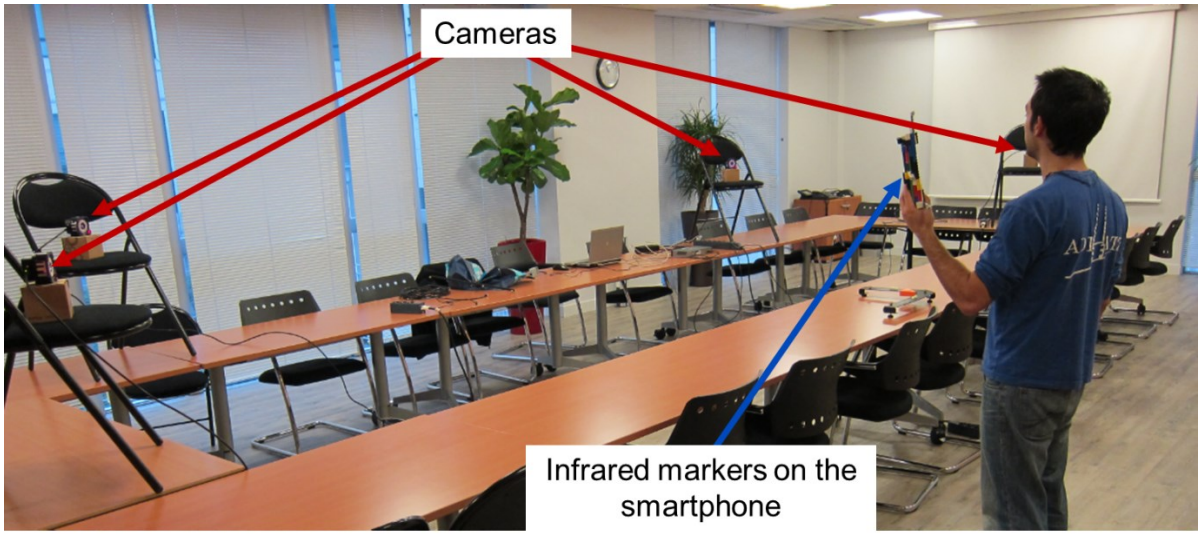


Figure 4-8: The setup used to record ground truth data. On top left the smartphone with infrared markers. Top right shows the infrared camera. Bottom shows the complete setup, with cameras and markers.

The main goal of this setup is to allow an accurate recording of the smartphone's motion while not disturbing the camera and inertial measurements. Every marker's position is triangulated using the cameras, and as we manually indicate that the markers all belong to a rigid body, the software records the pose of the platform. This is why we built a Lego© based shell where the smartphone is contained, and that infrared markers are fixed on that shell. The main issue is that at least four markers need to be in sight of at least three cameras at every moment. This limits the spatial coverage of the setup.

As the pose of the platform is recorded by two separate devices, a temporal calibration step had to be performed to align the two estimations. A weighted least square method was applied to calibrate temporally the two devices. The two estimations are used as inputs, and we look for the temporal shift and drift between the measurements. The M-estimator (3.1.1.2) using a bi-weight function typically converges in ten to twenty iterations for every sequence tested. A very similar method is used in [Karpenko et al. 2011] to calibrate the visual and inertial measurements with a gradient descent technique.

4.4.2 Tested methods

To fairly compare several methods of localization, the same input data need to be provided to each technique. In every sequence, a 3D map is initialized with a 7-point RANSAC algorithm. The only requirement is that the user performs a translation motion at the beginning of the sequence (otherwise, the 7-point algorithm offers poor results). For visual inputs, FAST/BRIEF is utilized for every algorithm except ours, where FAST/IBRIEF is used.

With the 3D map initialized, each technique performs its own odometry. The map is refined with an offline bundle adjustment, occurring about every 50 frames. It provides refined 3D map and poses, based on previous observations. As the comparison is made solely on the on-

the-fly odometry quality of the tested approaches, refined poses are not utilized to compare the methods. However, rectified 3D points are used.

Main inertial-visual odometry algorithms can be separated in two categories: determinists optimization-based and probabilistic filtering-based. A comparison is presented here between five inertial-visual approaches: three state of the art ones, and two MLFO-based techniques.

- **Optimization based method**

For the optimization-based category, an implementation of the **Iterative Weighted Least-Square** was applied, with a similar parameterization as the one presented in [Klein & Murray 2007]. A classical model of projection $CamProj$ is applied to compute the error in the M-estimator, which corresponds to the 3D point projection on a 2D plane: $I_x = KPX$ (see section 1.4.1 for further details). In our test model we do not incorporate the radial distortion correction.

The motion model used is based on the exponential map on $SE(3)$ that is 3D rotation and position representation (see appendix B.2 for more details on exponential map). To integrate inertial measurements, the pose utilized for the first iteration of the algorithm is a propagation of the previously estimated pose by the inertial sensors.

- **Filtering techniques**

Filtering-based algorithms are much more varied, therefore two of them were implemented as test references. The first one is an **Unscented Kalman Filter** (UKF), implemented and parameterized as in [Holmes et al. 2009] using the Square Root UKF. The second one is based on the **particle filter**, as in the position estimation procedure demonstrated in 4.3.2 , but this time applied for the whole 6D pose. We use a state vector that is composed of the 3D pose as a six items vector: three for rotation, three for position. It is exactly the same model as in optimization-based techniques.

Both state of the art techniques are only based on visual measurements. To integrate inertial measurements, we apply their recordings as the propagation step, as in [Aksoy & Alatan 2014]. The inertial sensors are not only used to propagate the state, but the covariance of the UKF (or the particles spreading) as well. This justifies the fact that we do not include the speed of rotation or translational speed in the motion model, as the propagation is performed with inertial sensors measurements. While an iterated method that is updated with each measurement is also widely used [Li & Mourikis 2013], we preferred to perform the fusion of both sensors at the same time, in order to have a more similar operation than the other methods.

- **Techniques developed in this thesis**

Our Multiple Level Fusion Odometry (section 4.3) algorithm is compared to the other algorithms. A first version with the classical BRIEF (MLFO w. BRIEF) and the second with IBRIEF (MLFO) are shown.

4.4.3 Sequences and results

To evaluate the quality of 3D odometry, orientation and position are separately compared, using the position err_{pos} absolute error (in centimeters) and rotation absolute error err_{rot} (in degrees) as first metrics.

About 14 sequences were recorded using this setup, in three different places. A variety of motion types has been performed in every place, to test the highest number of possible conditions. Figure 4-9 shows some frames extracted from sequences, to illustrate the recorded footage.

Figure 4-9: Frames extracted from some sequences.

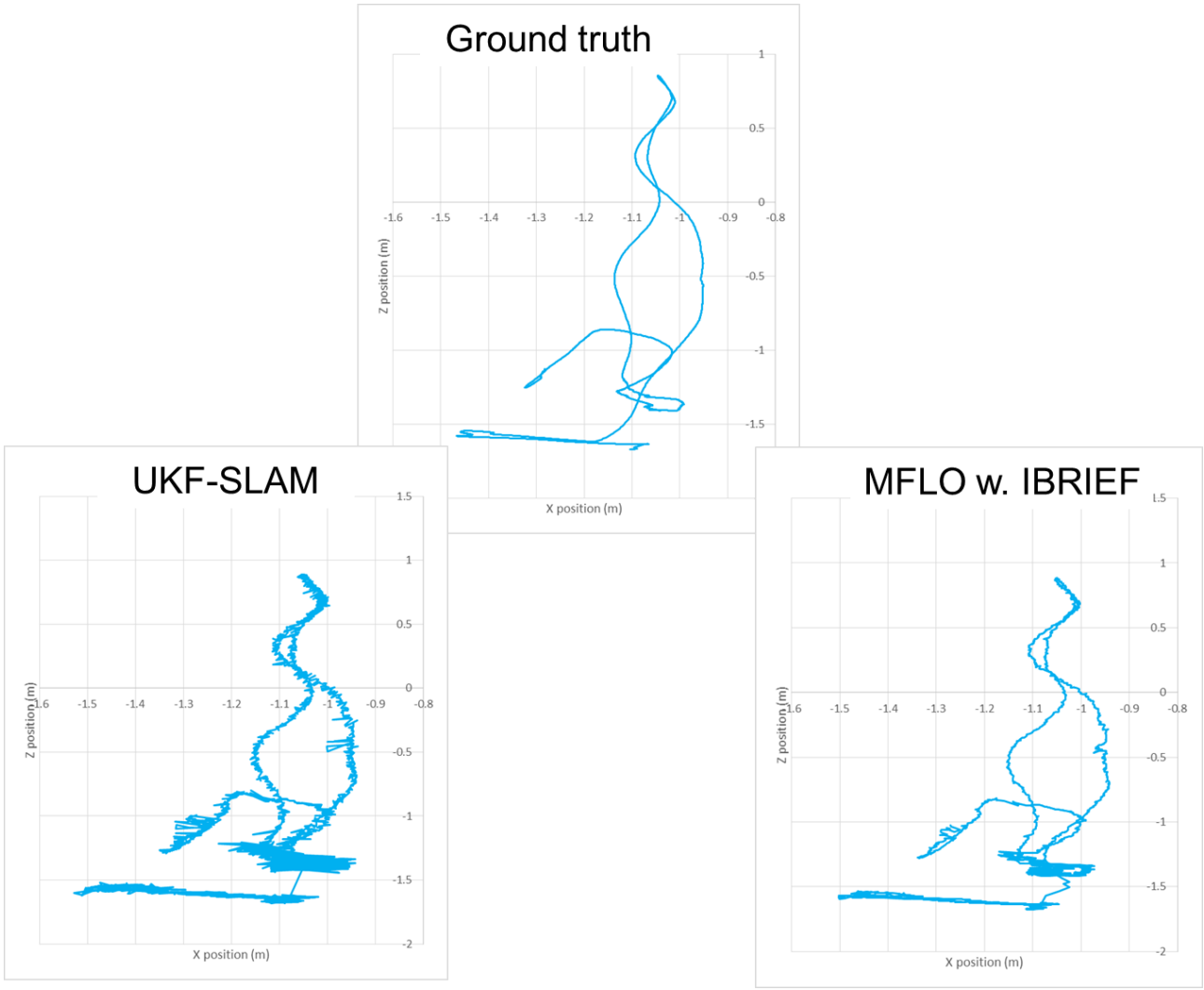


Figure 4-10: The x-z trajectories. The top one was recorded by the setup, bottom left corresponds to te UKF-SLAM estimation, while bottom right is the MLFO with IBRIEF estimation.

- **First sequence**

Two sequences that we believe are representative of the overall performance will be detailed in terms of results and errors, then a quick summary on the rest of the sequences will be shown. The first sequence is a "simple" forward-backward trajectory. The trajectory recorded by our infrared based setup, as well as some result curves are shown on Figure 4-10.

The scene can be seen on the left frame of Figure 4-9. As the camera orientation was kept more or less constant during the sequence, each method was more or less equivalent to the others in terms of orientation error. For position, Figure 4-11 displays the error of every method.

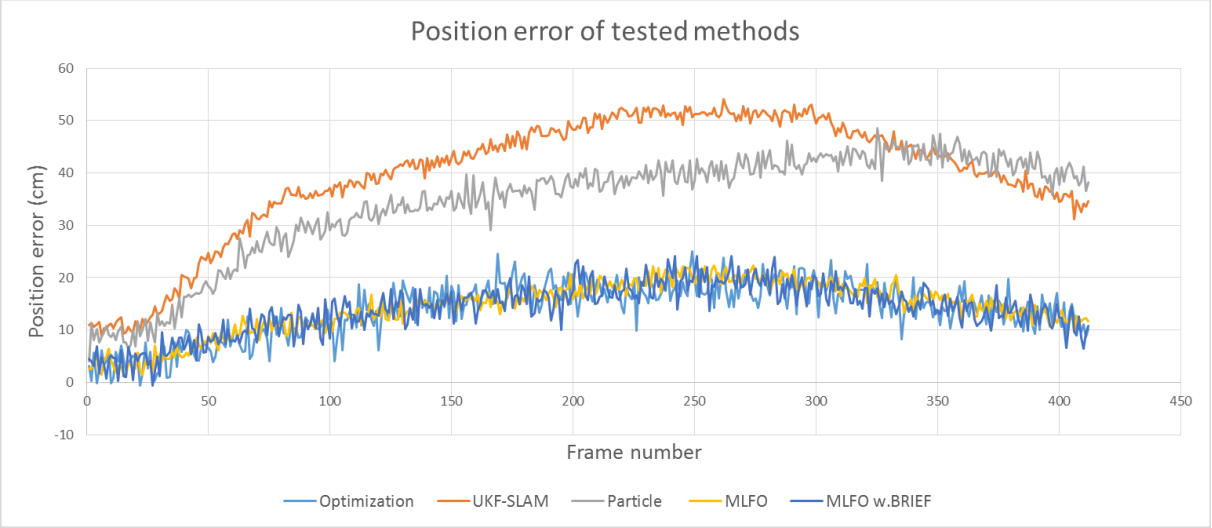


Figure 4-11: Error position of every tested method in the first sequence where the motion is a backward forward motion.

One can see on Figure 4-11 that the optimization and both MLFO techniques perform very similarly. UKF-SLAM is much more prone to errors, particle filtering being slightly below, even if for the end of the sequence the UKF-SLAM effectively reduces the error, as the sequence comes back to already seen objects. Particle filtering seems a bit slower to reduce the error regarding that. Another interesting feature is that filtering methods tend to display a better stability of error, whereas the optimization based technique seems less stable. For our approach, the error is generally on par with the optimization technique, even if the median error is slightly above (16 for MLFO vs. 15 for optimization, see Figure 4-13). For this sequence that one could qualify as "simple", no method was completely lost. While it is not completely demonstrable, we firmly believe that the difference between the filtering techniques and the others is the lack of direct outlier rejection for filtering based methods.

- **Second sequence**

For the second sequence we want to detail, a more complex motion was performed. In particular, some quick rotations occur in the sequence, while keeping the scene in the field of view. This underlines the resilience of the approaches to these types of motions that are quite frequent for handheld devices. Figure 4-12 shows the orientation error for the sequence. A similar trend for filtering techniques is seen as in the first sequence. Indeed, their lack of outlier rejection seems to induce a lesser robustness to tough contents. In this case, the importance of IBRIEF keypoints can also be seen, as the MLFO technique displays better performance than MLFO with BRIEF descriptors. The optimization method seems to perform similarly to MLFO except for very complex contents, where the hybrid RANSAC seems to better handle these cases.

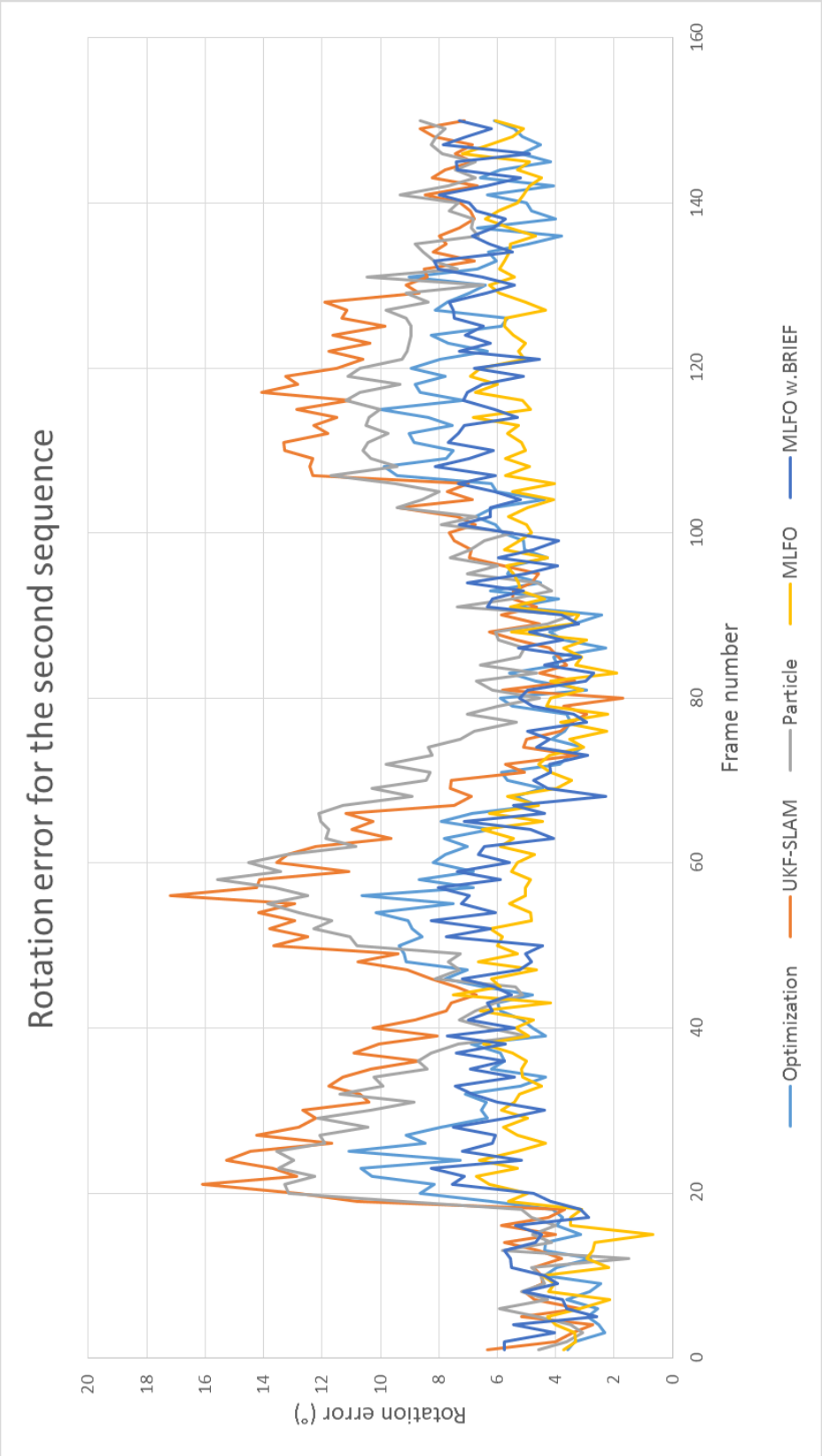


Figure 4-12: Orientation error on the second sequence for the tested methods.

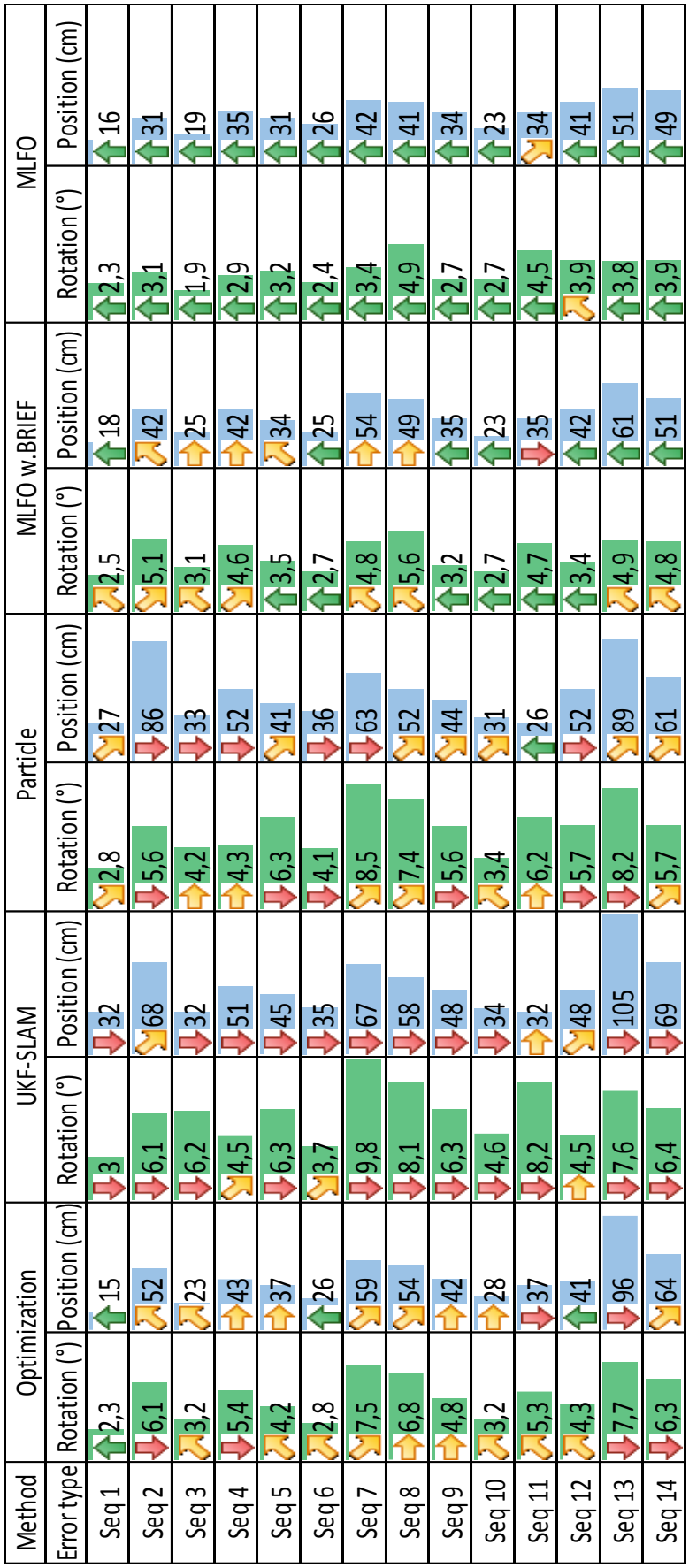


Figure 4-13: Table of the median error of position and orientation for every sequence and every method.

For the other sequences, Figure 4-13 displays the results of every method median error for the position and rotation: the larger the bar, the higher the error. Arrows close to the number indicates the relative performance of every method for the sequence. The more red (and going down) is the arrow, the highest error for this sequence, the greener the arrow (and going up), the lowest the error. Each arrow type represents a 20% slice of the error.

MLFO has often the lowest error of all methods, while UKF SLAM is often among the worst ones. Optimization-based technique performs well on most of the sequences. It can be seen that the addition of IBRIEF improves the result of our method, especially the ones with high rotations.

4.4.4 Complexity of the methods

Every method has different scaling regarding various parameters and the amount of points being tracked currently. In this section we state the complexity and computational time of every method tested. We remind that to keep every approach on the same level, BA is considered as the mapping technique for every one of them. Therefore only the localization stage is considered. We also wanted every method to display similar processing time, in order to place them on an equal footing.

- **UKF-SLAM**

Unscented Kalman filtering does not offer any parameters that will influence greatly the computational time. The dominant computational cost of a Kalman filter generally depends on the state vector size. In our experiments, this size is quite low as we only perform filtering on the motion model that is a six-element vector.

The cost of propagation is limited, as the state vector size is low, and the propagation is performed with inertial sensors measurements. Unscented transform (an inverse transform) costs are linear with the size of the state vector.

The dominant cost of the UKF is therefore in the correction step, where every sigma point is projected onto the measurements space. Each sigma point generates N 2D projection of the 3D points, where N is the number of visible points. Then the major cost is the matrix inversion of the mutual information, which leads to a computational load of $O(N^{2.4})$ [Thrun 2002]. In effect, the inverse unscented transform cost is only of $O(\mathrm{H} * N)$, H being the size of the state vector.

- **Particle filter**

The complexity of the particle filter depends mainly on two inputs: the amount of points N and the number of particles L. The cost of propagation of the particles is linear with respect to L.

The main cost is located in the computation of the error of each particle, where the reprojection error is computed for every point in every particle; this stage's complexity is $O(N * L)$.

It should be noted that the amount of particles impacts heavily the performance of the filter: the more particles, the better the performance. It should also be noted that the dimension of the state in this particle filter (six) is much more important than the one presented in section 4.3.2 (three). Therefore much more particles are needed.

In our experiments, we found that the filter was working decently at a number of about two thousand particles for the full pose estimation.

- **Optimization based**

The computational cost in optimization based method depends on two values: the number of points N and the number of iterations performed IT. The main source of computation is the calculus of the error and Jacobian matrix for each point. The update itself in the Levenberg-Marquart algorithm for an iteration is a matrix computation of type:

$$(JJ^T + \lambda I)\mu_u = J^T[\omega(r_i)r_i] \quad (55)$$

where J is the Jacobian matrix, λ the damping parameter, $\omega(r_i)r_i$ the weighted residuals of the correspondences, and μ_u the update to be added to the current state. This computation is also linear with respect to N. The procedure is stopped when the update is low enough.

Therefore the global complexity of the approach is of the type $O(N * IT)$. In practice, we limit the amount of iterations to fifteen to keep the processing low enough.

- **MLFO**

As the algorithm is a two-stage procedure, we study the complexity of each stage. For the rotation computation, the hybrid RANSAC complexity depends on two values: the amount of models M and the block size B, further details in section 3.4.2 . The complexity is $O(M * B)$. In our experiments, we found that a correct performance was reached at $M = 100$ and $B = 10$. These values are not very high due to the fact that the motion model is the 3D rotation that is computed on a simplified model.

For the particle filtering part, the complexity is also of $O(L * N)$, with a slight nuance. In effect, the particle filtering is only performed on the inliers that are not rejected by the hybrid RANSAC procedure. While the complexity of particle filtering is of the type $O(L * N)$, the real computational type is inferior as only inliers are considered. The total complexity of our approach is $O(M * B) + O(L * N)$. In our tests, a value of $L = 400$ and above allowed us to reach correct performance.

- **Computational time**

To illustrate the computational load needed for the methods to compute the pose with the given parameters, Figure 4-14 shows the measurements of processing time for each method, with a map of seven hundred points. Performances of a direct implementation with no specific algorithmic optimization were measured on a 2.8GHz double core, 4GB RAM computer. One can see that they are on a nearly equal footing, as intended. Overall, each method but the UKF can scale on some parameter to adapt its computational time.

Method	Optimization	UKF-SLAM	Particle	MLFO
Computational time (ms)	52	49	51	47

Figure 4-14: Processing time needed for each method tested.

4.4.5 Conclusions

We presented a novel approach to visual-inertial odometry, which constitutes the localization step in SLAM. Our method makes use of visual inertial fusion at various levels of the algorithm. Visual features are made more robust to rotational variations with inertial readings. The presented odometry approach mixes optimization techniques and probabilistic filtering. A specifically adapted Hybrid RANSAC is applied to compute 3D rotation, while also deleting outliers. Finally, position is computed thanks to a particle filter method that is highly adaptable to the matching quality, while also allowing several types of inputs. Ground-truth sequences were recorded, showing that our approach outperforms state of the art hybrid methods, especially in presence of blur or texture-less sequences.

Chapitre 5: Conclusion

Below is a French summary of chapter 5: Conclusion & perspectives.

La fusion de données capteurs inertiels et caméras est un problème complexe, offrant de multiples choix techniques. Cette thèse a présenté plusieurs méthodes afin d'accomplir cette tâche.

Un état de l'art sur les techniques de calcul de vecteur de mouvements a été montré au lecteur. Cette étape est généralement la première dans les algorithmes d'estimation de mouvement. Nous avons concentré notre attention sur les méthodes les moins couteuses en temps de calcul, les détecteurs simples et les descripteurs binaires. Une amélioration géométrique des descripteurs via les mesures inertielles a été proposée, ajoutant l'invariance aux rotations planaires et améliorant la robustesse aux changements de point de vue. Nous avons montré que les améliorations sont significatives, et faites avec un surcout de calcul très faible.

Une revue des techniques robustes d'estimation de mouvement 2D a été présentée. Les algorithmes visuels et inertiels ont été étudiés. Les procédures de type RANSAC sont particulièrement intéressantes pour leur résilience aux vecteurs de mouvement incorrects. Nous avons proposé un algorithme de RANSAC hybride, qui intègre la fusion au cœur de la procédure en utilisant un score hybride des modèles. Cette technique a été améliorée avec l'ajout d'un modèle inertiel et un lagrangien dynamique. Des tests ont été conduits afin de démontrer l'efficacité de l'approche.

Les méthodes de SLAM ont été étudiées, tant celles se reposant uniquement sur la vision que les approches hybrides visuelles-inertielles. Après avoir mené des expériences pratiques sur les méthodes existantes, nous obtenons une meilleure compréhension des avantages et inconvénients des approches à base de filtrage et d'optimisation. Une nouvelle approche de fusion à des multiples niveaux dans l'algorithme est introduite. Grâce à un système de caméras et de marqueurs infrarouges, nous obtenons une vérité terrain du mouvement de l'appareil, qui permet de démontrer que l'approche proposée est supérieure aux méthodes de l'état de l'art.

Chapter 5: Conclusion & perspectives

5.1 Conclusions

Fusion between inertial and visual sensors for motion estimation in embedded video application is a complex task to perform, with many possible choices of techniques. This thesis has presented some techniques to solve this problem.

We have provided the reader with an overview on the visual feature estimation techniques, which are generally the first step of most motion estimation algorithms. We focused our work on the least expensive techniques existing in the literature, simple detectors and binary descriptors. A geometric robustness improvement, based on the inertial sensors reading, has been proposed, which not only provides in-plane rotation invariance, but also increased resilience to viewpoint changes. We have shown that the robustness improvement was significant, but also performed at very little additional cost, thanks to specific parameterization and approximations.

A review on the robust techniques of 2D motion estimation was presented. Hybrid visual inertial systems have been studied, with a focus on RANSAC procedure, which displays a very high robustness to outliers. A hybrid RANSAC algorithm has been designed, in order to integrate the inertial measurement at the heart of the procedure, thanks to a hybrid scoring of the motion models. This technique was also improved with an inertial-based model especially relevant in case of very complex scenes, as well as a dynamic behavior of the procedure, to adapt itself to the complexity of the scene. Tests were conducted to demonstrate the efficiency of the approach.

We investigated SLAM techniques, from pure visual ones to hybrid visual inertial ones. Hands-on experiments led to a better understanding of the advantages and drawbacks from state of the art techniques, which either use filtering or optimization strategies. We proposed a novel technique performing fusion in many stages of the algorithm. A hybrid RANSAC is applied to estimate the orientation of the device, also removing outliers from the motion vectors set. Then, a particle filter is performed to compute the position of the platform, with dynamic parameterization as well as proposing the possibility of integrating semantic information in the filter. A setup using infrared markers and cameras has been designed and used in order to record very accurately the motion of the platform. These recordings were used as ground truth data to compare our approach with state of the art ones. Our method outperformed the state of the art ones, thanks to a deeply integrated fusion between inertial and visual measurements.

5.2 Perspectives

We believe that the interaction between visual and inertial measurement could be increased for keypoint techniques. Robustness to artifacts such a motion blur can probably be improved using inertial sensors. The description method could be dynamically modified with the inertial readings in order to cope with rolling shutter distortions. Some techniques such as expectation maximization ones could be used to cope with the keypoint changes of appearance, rather than just the viewpoint change, in order to offer a better reactivity to fast changing keypoints (internal motions for instance).

2D motion estimation using the hybrid RANSAC could be extended to more complex motion models, such as some dealing with rolling shutter, perspective changes, etc… Specific adaptations in terms of motion models as well as scoring function should be designed in order to better handle these cases. A remaining issue in the hybrid RANSAC is that a total outlier in the inertial sensors reading (magnetic distortions for instance) heavily affects the procedure. A mechanism providing control on the inertial readings should be integrated to the algorithm, to increase robustness.

Our Multiple Level of Fusion Odometry approach can be improved in many domains. More semantic types of data can be taken as input for the position estimation filter. A dynamic setting of the algorithm parameters (number of motion models for hybrid RANSAC, particle number) could be used to reach maximum efficiency. The initialization method does not necessitate any assumption on the scene, but some type of motion is required (mostly translation one). This could be improved, to provide a quicker and less constrained initialization phase. We know that relocalization in case of completely lost tracking can be performed with PnP techniques, but we believe that this specific stage of the SLAM could be studied, with faster re-initialization or a particular mode activation in case of very poor tracking.

Publications

Conferences

Alibay Manu, Auberger Stephane, Stanciulescu Bogdan, Fuchs Philippe. 2014. Hybrid visual and inertial RANSAC for real-time motion estimation. In *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, pp. 179–183.

Patents

Alibay Manu, Auberger Stephane. Visual-inertial fusion motion estimation in the preemptive RANSAC framework. Filed 14 Jan. 2014.

Alibay Manu, Auberger Stephane. Specific particle filter parameterization. Proposed 9 June 2015, to be filed soon.

Alibay Manu, Auberger Stephane. Adaptative error threshold to point distance. Proposed 9 June 2015, to be filed soon.

Alibay Manu, Auberger Stephane, Stanciulescu Bogdan, Fuchs Philippe. Hybrid BRIEF. Proposed 9 June 2015, to be filed soon.

Alibay Manu, Auberger Stephane, Stanciulescu Bogdan, Fuchs Philippe. Splitting rotation and position estimation in SLAM. Proposed 9 June 2015, to be filed soon.

Chapter 5: Conclusion & perspectives

Appendix A: Kalman Filtering

Kalman filter, proposed by R.E. Kalman in 1960 [Kalman 1960], is an estimation method that makes use of state representation and state transition to propagate a linear system obtaining optimal estimation error, and can be seen as an extension of the Wiener filtering theory.

A.1 Classical Kalman filter

It is used to combine two estimates of a quantity optimally in order to find an estimate with minimal uncertainty. Kalman filter makes use of the state-space representation. This representation takes the output of the system as system states. Because of their similarities, it can be considered as a state-space version of Recursive Least Squares (RLS) filtering.

Kalman filter propagates a stochastic state - the state is represented by a probability density function. This probability density function is parameterized as a Gaussian distribution and described by a mean vector and a covariance matrix.

As discussed above, Kalman filter is used to combine two full or partial estimates of the state vector. Each iteration of a Kalman filter consists of two stages: state transition or prediction and measurement update or correction.

In the **state transition stage**, a hypothesis for the next state is generated. Previous state vector and its covariance are the input of this stage with optional user control input vector. Kalman filter requires a linear prediction function:

$$\mu_n = A_n\mu_{n-1} + B_n u_n + \varepsilon_n \tag{56}$$

Where μ_{n-1} and μ_n represent the previous and current estimated state vectors, u_n is the control input vector, and ε_n is a zero-mean Gaussian random vector modeling the uncertainty introduced by state transition with covariance Γ. A_n and B_n are matrices that maps the previous state and the control input to the next state, respectively.

The **measurement update stage** combines the hypothesis generated in the state transition stage and combines it with an observation of the state, called measurement. It is also characterized by a linear function:

$$y_n = C_n\mu_n + \eta_n \tag{57}$$

Where y_n is the measurement vector, η_n is a zero-mean Gaussian random vector modeling the uncertainty in the measurement. C_n is a matrix that maps the state to the measurement vector. Since the defined functions are linear in their arguments and the arguments are Gaussian random vectors, the estimated new state is also a Gaussian.

The prediction for the next state and its covariance is computed as:

$$\hat{\mu}_n = A_n\mu_{n-1} + B_n u_n \tag{58}$$
$$\hat{M}_n = A_n M_{n-1} {A_n}^T + \Gamma_n \tag{59}$$

Appendix A: Kalman Filtering

M represents the covariance of the state vector and the variables with represents that they are prediction of the regular ones. The power of Kalman filter lies in the Kalman gain. Kalman gain (κ_n) specifies how the prediction $\hat{\mu}_n$ and the measurement y_n are combined to get the next state vector μ. It is computed as:

$$\kappa_n = \widehat{M}_n C_n{}^T \left(C_n \widehat{M}_n C_n{}^T + \Phi_n\right)^{-1} \tag{60}$$

Then the predicted state is updated with the measurements, weighted by the Kalman gain:

$$\mu_n = \hat{\mu}_n + \kappa_n (y_n - C_n \hat{\mu}_n) \tag{61}$$

Finally, the covariance is computed for the current state:

$$M_n = (I - \kappa_n C_n)\widehat{M}_n \tag{62}$$

A.2 Extended Kalman filter

The Kalman filter is restricted to linear systems and measurements regarding the state, due to its matrix-based formulation. In many cases this restriction heavily limits the application of this technique, as a lot of systems are not linear. To overcome this issue, the Extended Kalman Filter (EKF) has been developed. It consists in linearizing the propagation and measurement function locally with a Taylor expansion of the model.

The propagation and measurement stages are now modeled with function g and h respectively:

$$\mu_n = g(\mu_{n-1}, u_n) + \varepsilon_n \tag{63}$$

$$y_n = h(\mu_n) + \eta_n \tag{64}$$

The Taylor expansion of the propagation equation is formulated as:

$$g(\mu_n, u_n) \approx g(\mu_{n-1}, u_n) + J_{g(\mu_{n-1}, u_n)}(\mu_{n-1})(\mu_n - \mu_{n-1}) \tag{65}$$

where $J_{g(\mu_{n-1}, u_n)}(x)$ is the Jacobian of function $g(\mu_{n-1}, u_n)$ with respect to x. A similar linearization is applied to the measurement function h.

The equations of the EKF are highly similar to the ones of the Kalman filter, with the linearization included:

$$\hat{\mu}_n = g(\mu_{n-1}, u_n) \tag{66}$$

$$\widehat{M}_n = J_{g(\mu_{n-1}, u_n)}(\mu_{n-1}) M_{n-1} J_{g(\mu_{n-1}, u_n)}(\mu_{n-1})^T + \Gamma_n \tag{67}$$

$$\kappa_n = \widehat{M}_n J_{h(\mu_{n-1})}(\mu_{n-1})^T \left(J_{h(\mu_{n-1})}(\mu_{n-1}) \widehat{M}_n J_{h(\mu_{n-1})}(\mu_{n-1})^T + \Phi_n\right)^{-1} \tag{68}$$

$$\mu_n = \hat{\mu}_n + \kappa_n (y_n - h(\hat{\mu}_n)) \tag{69}$$

$$M_n = (I - \kappa_n J_{h(\hat{\mu}_n)}(\hat{\mu}_n))\widehat{M}_n \quad (70)$$

A.3 Unscented Kalman filter

The Unscented Kalman Filter (UKF) was introduced in [Julier & Uhlmann 1997], and makes use of an Unscented Transform (UT) to overcome the limitation to linear systems of the Kalman filter. The UT consists in transforming a Gaussian by a finite number of sigma points, which are particles in the state space. An inverse transformation (that is noted UT^{-1}) is presented to recompute the mean and covariance of the state. This inverse transformation is exact in the case of a Gaussian probability density.

The dimension of the state vector is here noted n. The UT generates $2n + 1$ sigma points. Weights are associated to each sigma point. These weights differ for the computation of the mean (noted ω_μ^i) and the covariance (noted ω_M^i).

- Unscented Transform

This operation is based on the covariance matrix M and the mean value μ of the distribution. The notation $(\sqrt{(n+\lambda)M})_i$ denotes the i-th line (or column) of the square root matrix of $(n + \lambda)M$. The sigma points $\chi(i)$ are computed as:

$$\begin{cases} \chi(0) = \mu \\ for\ i\ \in [1,n], \chi(i) = \mu + (\sqrt{(n+\lambda)M})_i \\ for\ i\ \in [n+1,2n], \chi(i) = \mu - (\sqrt{(n+\lambda)M})_{i-n} \end{cases} \quad (71)$$

With their associated weights for the computation of mean and covariance:

$$\omega_\mu^i = \frac{\lambda}{n+\lambda} \quad (72)$$

$$\omega_M^i = \frac{\lambda}{n+\lambda}(1 - \alpha^2 + \beta) \quad (73)$$

where λ is a scale parameter that is set to $n - 3$ in case of a Gaussian distribution. The parameters α, β can be adapted to model several distributions. This is an advantage of the UKF, as it can be parameterized to support other distribution. The EKF is quite robust to other distributions, but cannot be specifically adapted like the UKF.

- Inverse Unscented Transform

We note the propagated sigma points as $\gamma(i) = g(\chi(i), u_n)$. The operation of computing the mean and covariance of the propagated distribution is then:

Appendix A: Kalman Filtering

$$\hat{\mu}_n = \sum_{i=0}^{2n} \omega_\mu^i \gamma(i) \tag{74}$$

$$\widehat{M}_n = \sum_{i=0}^{2n} \omega_M^i (\gamma(i) - \hat{\mu}_n)(\gamma(i) - \hat{\mu}_n)^T \tag{75}$$

- Propagation step

The propagation simply consists in performing the UT of the previous distribution, applying the function g to the sigma points, and performing a UT^{-1} to compute the predicted mean and covariance.

$$\{\chi_{n-1}(i), \omega_{\mu_n}^i, \omega_{M_n}^i\} = UT(\mu_n, M_n) \tag{76}$$

$$\gamma_{n-1}(i) = g(\chi_{n-1}(i), u_n) \tag{77}$$

$$\{\hat{\mu}_n, \widehat{M}_n\} = UT^{-1}(\gamma_{n-1}(i), \omega_{\mu_n}^i, \omega_{M_n}^i) + \Gamma_n \tag{78}$$

- Measurement update

The measurement update consists in performing a UT on the predicted distribution, projecting these sigma points into the measurement space. Then these projected sigma points are transformed back into a mean covariance representation, which is used to compute the Kalman gain along a measurement covariance M_z.

$$\{\hat{\chi}_n(i), \omega_{\hat{\mu}_n}^i, \omega_{\widehat{M}_n}^i\} = UT(\hat{\mu}_n, \widehat{M}_n) \tag{79}$$

$$Z_n(i) = h(\hat{\chi}_n(i)) \tag{80}$$

$$\{z_n, \hat{S}_n\} = UT^{-1}(Z_n(i), \omega_{\hat{\mu}_n}^i, \omega_{\widehat{M}_n}^i) + \Phi_n \tag{81}$$

$$M_z = \sum_{i=0}^{2n} \omega_{\widehat{M}_n}^i (\hat{\chi}_{n-1}(i) - \hat{\mu}_n)(Z_n(i) - z_n)^T \tag{82}$$

The Kalman gain is computed by taking into account the covariance from the predicted measurements and the measurement covariance:

$$\kappa_n = M_z \hat{S}_n^{-1} \tag{83}$$

Finally, the new state is estimated exactly as in the EKF, and the covariance is computed as:

$$M_n = \widehat{M}_n - \kappa_n \hat{S}_n \kappa_n^T \tag{84}$$

The handling of non-linear transformations in the UKF is applied to the sigma points, either in the propagation or measurement update, before re-computing the probability density with its mean and covariance. The UKF possesses mainly two advantages compared to the EKF. Firstly, the Jacobian is not utilized, therefore we do not need to calculate it, which makes the implementation much more flexible to changes in the model. Secondly, the UKF handles highly

non-linear systems better than the EKF [Julier & Uhlmann 1997]. The main drawback of the UKF is that its computation cost can be slightly higher than the one of the EKF if the Jacobian computation is not too expensive. However, for large dimensions, the predominant cost of both approaches is the same, and lies in the inversion of the matrix in the computation of the Kalman gain.

Appendix A: Kalman Filtering

Appendix B: 3D rotation representations

Many representations for three dimensional rotations exist. We here present two of them that are often used in the motion estimation literature: quaternions and exponential maps.

B.1 Quaternions

Quaternions possess a lot of mathematical history and backgrounds. They constitute a group based on the four dimensional vector set R^4 with specific operator for multiplication '$\circ$'. The sub-group S^3 corresponds to quaternions of unit length, and is used to represent rotations. A quaternion $q = [q_x, q_y, q_z, q_w]^T$ encodes a rotation of angle θ around a unit 3D axis μ as:

$$[q_x, q_y, q_z, q_w]^T = [\mu \sin\left(\frac{\theta}{2}\right), \cos\left(\frac{\theta}{2}\right)]^T \qquad (85)$$

Formulas used to compute a rotation matrix R and its partial derivates based on quaternions members are introduced in [Shoemake 1985]. They are not linearly dependent over S^3. This means that unit quaternions are free from the gimbal lock issue that is present in Euler angles. As for rotation matrices, quaternions possess more members (4) than the degree of freedom of a 3D rotation (3). When performing filtering or optimization on quaternions representing rotations, one must ensure that their length is kept unit. This is very simply done by dividing each member of the quaternion by the total norm of it, which is much simpler than the orthonormalization required by rotation matrices.

Quaternions have numerous advantages over Euler angles and direct rotation matrices when it comes to representing rotations, they offer a good balance between numerical properties and do not present artifacts. They are widely used for inertial sensor fusion [Sabatini 2006] [Wheeler & Ikeuchi 1995].

B.2 Exponential maps

Any 3D rotation can be expressed as a rotation of an angle θ around a unit 3D axis μ. Therefore, a relationship between this representation and the rotation matrix R that describes the same 3D rotation has to be demonstrated. It is shown in [Murray et al. 1994] that, when performing this task, one ends up with:

$$R(\theta, \mu) = e^{\hat{\mu}\,\theta} \qquad (86)$$

Where $\hat{\mu}$ is the skew symmetric matrix of the 3D axis $\mu = [\mu_x, \mu_y, \mu_z]^T$:

Appendix B: 3D rotation representations

$$\hat{\mu} = \begin{pmatrix} 0 & -\mu_z & \mu_y \\ \mu_z & 0 & -\mu_x \\ -\mu_y & \mu_x & 0 \end{pmatrix} \tag{87}$$

The computation of $e^{\hat{\mu}\,\theta}$ is then based on the Taylor expansion:

$$e^{\hat{\mu}\,\theta} = I + \hat{\mu}\,\theta + \frac{(\hat{\mu}\,\theta)^2}{2!} + \frac{(\hat{\mu}\,\theta)^3}{3!} + \cdots \tag{88}$$

By applying properties of skew symmetric matrices of unit vectors, the formula is split in two parts that correspond to expansions of sinusoid functions. This leads to the Rodrigues formula:

$$R(\theta, \mu) = e^{\hat{\mu}\,\theta} = I + \hat{\mu}\,\sin(\theta) + \hat{\mu}^2(1 - cos(\theta)) \tag{89}$$

The main advantage of the exponential map is its ease of differentiation due to the exponential function. More details on exponential map and Lie algebra can be found in [Varadarajan 1974]. It should also be noticed that, as in this representation all angles are applied simultaneously in the computation of the rotation matrix, there is no gimbal lock nor singularities.

Appendix C: Particle filter

Particle filtering relies on the law of the big numbers, and is also known as the sequential method of Monte-Carlo. This technique consists in estimating a function based on samples to estimate it:

$$\int g(x)dx = \lim_{N\to\infty} 1/N \sum_{i} g(i) \tag{90}$$

The main advantage of this technique is that it does not suffer from any flaw of the usual filtering techniques, as long as the number of samples that we can use is high enough. Estimating a function using many samples can be done to perform many tasks: learning algorithms, computing the best possible solution given input data (ground truth generation), etc… When done sequentially, its main domains of application in computer vision are tracking and filtering.

The algorithm proceeds in three steps: propagation, measurement and diffusion. The propagation consists in applying a temporal model to the particles:

$$\forall l \in \{1, \dots, L\}, \xi_l^t = f(\xi_l^t) \tag{91}$$

with L being the total number of particles. Then, the probability of each particle is measured with a function π. This function is used to produce weights that allow the estimation of the state:

$$\omega_l^t = \frac{\pi\left(\xi_l^t \middle| v(1, \dots, N)\right)}{\pi_{tot}} \tag{92}$$

where π_{tot} is the sum of all probabilities $\pi(\xi_l^t)$ for each particle at time t. Then the estimated state is simply the weighted sum of the particles:

$$\mu_t = \sum_{l=0}^{L} \omega_l^t \xi_l^t \tag{93}$$

Then, the diffusion step consists in producing new particles in order to continue the procedure. This is mainly the stage that makes particle filtering a very rich domain, as many strategies exist to perform this. The particle swarm method is described in section 4.3.2 , where the best particle is used in combination with a random vector to diffuse the particles. Importance sampling diffuses the particles using previous ones, preferably selecting the ones with higher

probabilities. Many other techniques can be utilized, depending on the filter context, computational time available, expected distribution type, etc…

Appendix D: References

Ake, B., 1990. *Numerical Methods For Least Squares Problems*,

Aksoy, Y. & Alatan, A.A., 2014. Uncertainty modeling for efficient visual odometry via inertial sensors on mobile devices. In *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, pp. 3397–3401.

Alahi, a., Ortiz, R. & Vandergheynst, P., 2012. FREAK: Fast Retina Keypoint. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp.510–517.

Alibay, M. et al., 2014. Hybrid visual and inertial RANSAC for real-time motion estimation. In *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, pp. 179–183.

Arth, C., Mulloni, A. & Schmalstieg, D., 2012. Exploiting Sensors on Mobile Phones to Improve Wide-Area Localization. *International Conference on Pattern Recognition*, (Icpr), pp.2152–2156.

Auberger, S. & Alibay, M., 2014. Rolling shutter wobble detection and correction.

Auberger, S. & Miro, C., 2005. Digital video stabilization architecture for low cost devices. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.* IEEE, pp. 474–479.

Azuma, R., Hoff, B. & Iii, H.N., 1998. Making augmented reality work outdoors requires hybrid tracking. ... *on Augmented Reality*, pp.1–6.

Bailey, T. et al., 2006. Consistency of the EKF-SLAM Algorithm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 3562–3568.

Baker, S. et al., 2010. Removing rolling shutter wobble. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp.2392–2399.

Baldwin, G. et al., 2007. Complementary Filter Design on the Special Euclidean Group SE(3). *Proc. of the European Control Conference*, 1(3), pp.3763–3770.

Battiato, S. et al., 2007. SIFT Features Tracking for Video Stabilization. In *14th International Conference on Image Analysis and Processing (ICIAP 2007)*. IEEE, pp. 825–830.

Bay, Tuytelaars, H.T., 2006. SURF : Speeded - Up Robust Features. , pp.1–30.

Ben-Ezra, M. & Nayar, S.K., 2003. Motion deblurring using hybrid imaging. *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, 1, pp.1–8.

Appendix D: References

Bergen, J.R. et al., 1992. Hierarchical model-based motion estimation. In G. Sandini, ed. *Computer Vision—ECCV'92*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 237–252.

Bleser, G. & Stricker, D., 2008. Advanced tracking through efficient image processing and visual-inertial sensor fusion. *2008 IEEE Virtual Reality Conference*, pp.137–144.

Bouguet, J.Y., 2001. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5, pp.1–10.

Bovik, A.C., 2010. *Handbook of Image and Video Processing*,

Brajdic, A. & Harle, R., 2012. Scalable indoor pedestrian localisation using inertial sensing and parallel particle filters. In *2012 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2012 - Conference Proceedings*. pp. 13–15.

Brown, M. & Lowe, D.G., 2006. Automatic Panoramic Image Stitching using Invariant Features. *International Journal of Computer Vision*, 74(1), pp.59–73.

Bruhn, A., Weickert, J. & Schnörr, C., 2005. Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods. *International Journal of Computer Vision*, 61(3), pp.1–21.

Calonder, M. et al., 2010. BRIEF : Binary Robust Independent Elementary Features. In *Proceedings of the European Conference on Computer Vision. ECCV'10*. pp. 778–792.

Calonder, M. et al., 2012. BRIEF: Computing a Local Binary Descriptor Very Fast. *IEEE transactions on pattern analysis and machine intelligence*, 34(7), pp.1281–98.

Capel, D.P., 2005. An Effective Bail-out Test for RANSAC Consensus Scoring. *Procedings of the British Machine Vision Conference 2005*, pp.78.1–78.10.

Castle, R.O. et al., 2007. Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, pp. 4102–4107.

Chum, O. & Matas, J., 2005. Matching with PROSAC - Progressive Sample Consensus. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp. 220–226.

Chum, O. & Matas, J., 2008. Optimal randomized RANSAC. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8), pp.1472–1482.

Civera, J. et al., 2010. 1-Point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27, pp.609–631.

Civera, J., Davison, A.J. & Montiel, J.M.M., 2008. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 24(October), pp.932–945.

Coakley, C.W., 1997. *Robust statistical procedures: Asymptotics and interrelations*,

Coughlan, J.M. & Yuille, A.L., 1999. Manhattan World: compass direction from a single image by Bayesian inference. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*. IEEE, pp. 941–947 vol.2.

Crassidis, J.L. & Markley, F.L., 2003. Unscented Filtering for Spacecraft Attitude Estimation. *Journal of Guidance, Control, and Dynamics*, 26(JULY 2003), pp.536–542.

Crawford, a J. et al., 2004. Projections for Real Time Video Stabilisation. *Electrical Engineering*, pp.3371–3374.

Davison, A.J. et al., 2007. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6), pp.1052–1067.

Davison, A.J., 2003. Real-time simultaneous localisation and mapping with a single camera. *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 2, pp.1403–1410.

Diel, D.D., DeBitetto, P. & Teller, S., 2005. Epipolar Constraints for Vision-Aided Inertial Navigation. In *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1*. IEEE, pp. 221–228.

Durrie, J. et al., 2009. Vision-aided inertial navigation on an uncertain map using a particle filter. In *2009 IEEE International Conference on Robotics and Automation*. IEEE, pp. 4189–4194.

Eade, E. & Drummond, T., 2006. Scalable Monocular SLAM Simultaneous Localization and Mapping.

Engel, J., Sturm, J. & Cremers, D., 2013. Semi-dense Visual Odometry for a Monocular Camera. In *2013 IEEE International Conference on Computer Vision*. IEEE, pp. 1449–1456.

Euston, M. et al., 2008. A complementary filter for attitude estimation of a fixed-wing UAV. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.340–345.

Faugeras, O.D. & Lustman, F., 1988. Motion and Structure From Motion in a Piecewise Planar Environment. *International Journal of Pattern Recognition and Artificial Intelligence*, 02(03), pp.485–508.

Feliz, R., Zalama, E. & García-Bermejo, J.G., 2009. Pedestrian tracking using inertial sensors. In *Journal of Physical Agents*. pp. 35–42.

Filliat, D., 2007. A visual bag of words method for interactive qualitative localization and mapping. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, pp. 3921–3926.

Appendix D: References

Fischler, M. a. & Bolles, R.C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), pp.381–395.

Forssen, P. & Ringaby, E., 2010. Rectifying rolling shutter video from hand-held devices. *Computer Vision and Pattern ….*

Förstner, W., 1994. *A framework for low level feature extraction* J.-O. Eklundh, ed., Berlin, Heidelberg: Springer Berlin Heidelberg.

Fox, J., 1997. *Applied regression analysis, linear models, and related methods.*,

Foxlin, E., 1996. Inertial head-tracker sensor fusion by a complementary separate-bias Kalman filter. *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*, pp.185–195.

Freeman, W.T. & Adelson, E.H., 1991. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13, pp.891–906.

Furukawa, Y. et al., 2009. Manhattan-world stereo. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp.1422–1429.

Fux, S., 2008. Development of a planar low cost Inertial Measurement Unit for UAVs and MAVs.

Gauglitz, S., Höllerer, T. & Turk, M., 2011. Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking. *International Journal of Computer Vision*, 94(3), pp.335–360.

De Haan, G. et al., 1993. True-motion estimation with 3-D recursive search block matching. *IEEE Transactions on Circuits and Systems for Video Technology*, 3, pp.368–379.

Hanning, G. et al., 2011. Stabilizing cell phone video using inertial measurement sensors. *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp.1–8.

Haralick, B.M. et al., 1994. Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13(3), pp.331–356.

Harris, C. & Stephens, M., 1988. A Combined Corner and Edge Detector. *Procedings of the Alvey Vision Conference 1988*, pp.147–151.

Harris, C.G. & Pike, J.M., 1988. 3D positional integration from image sequences. *Image and Vision Computing*, 6(2), pp.87–90.

Hartley, R., 2006. Five-Point Motion Estimation Made Easy. *18th International Conference on Pattern Recognition (ICPR'06)*, pp.630–633.

Hartley, R. & Zisserman, A., 2003. *Multiple View Geometry in Computer Vision*,

Hartley, R.I., 1997. In defence of the 8-point algorithm. *Proceedings of IEEE International Conference on Computer Vision*, pp.1064–1070.

Holmes, S. a., Klein, G. & Murray, D.W., 2009. An O(N2) square root unscented kalman filter for visual simultaneous localization and mapping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7), pp.1251–1263.

Horn, B.K.P. & Schunck, B.G., 1981. Determining optical flow. *Artificial Intelligence*, 17, pp.185–203.

Hwangbo, M., Kim, J.-S. & Kanade, T., 2009. Inertial-aided KLT feature tracking for a moving camera. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.1909–1916.

Im, J., Kim, D. & Hong, K., 2006. Digital Video Stabilization Algorithm for CMOS Image Sensor. In *2006 International Conference on Image Processing*. IEEE, pp. 3261–3264.

Jia, C. & Evans, B.L., 2012. Probabilistic 3-D motion estimation for rolling shutter video rectification from visual and inertial measurements. *2012 IEEE 14th International Workshop on Multimedia Signal Processing, MMSP 2012 - Proceedings*, pp.203–208.

Julier, S.J. & Uhlmann, J.K., 1997. A New Extension of the Kalman Filter to Nonlinear Systems. *System Identification*, 3, pp.3–2.

Kalman, R.E., 1960. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME-Journal of Basic Engineering*, 82(Series D), pp.35–45.

Kang, S.B. et al., 2003. High dynamic range video. In *ACM Transactions on Graphics*. New York, New York, USA: ACM Press, pp. 319–325.

Karpenko, A. et al., 2011. Digital Video Stabilization and Rolling Shutter Correction using Gyroscopes. In *Cstr*. pp. 1–7.

Ke, Y.K.Y. & Sukthankar, R., 2004. PCA-SIFT: a more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* IEEE, pp. 506–513.

Kim, J.S., Hwangbo, M. & Kanade, T., 2009. Realtime affine-photometric KLT feature tracker on GPU in CUDA framework. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops 2009*. IEEE, pp. 886–893.

Kiriy, E. & Buehler, M., 2002. Three-state Extended Kalman Filter for Mobile Robot Localization Image Processing at CMU.

Klein, G. & Drummond, T., 2003. Robust visual tracking for non-instrumental augmented reality. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.* IEEE Comput. Soc, pp. 113–122.

Appendix D: References

Klein, G. & Murray, D., 2007. Parallel Tracking and Mapping for Small AR Workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp.1–10.

Klein, G. & Murray, D., 2009. Parallel Tracking and Mapping on a camera phone. *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pp.83–86.

Klein, G.S.W. & Drummond, T.W., 2004. Tightly integrated sensor fusion for robust visual tracking. In *Image and Vision Computing*. pp. 769–776.

Kneip, L., Scaramuzza, D. & Siegwart, R., 2011. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *CVPR 2011*. IEEE, pp. 2969–2976.

Koga, T. et al., 1981. Motion compensated interframe coding for video conferencing. *Proc. Nat. Telecommun. Conf., New Orleans*, pp.G5.3.1–5.3.5.

Kurz, D. & Benhimane, S., 2011. Gravity-aware handheld augmented reality. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011*. pp. 111–120.

Kurz, D. & Benhimane, S., 2012. Handheld augmented reality involving gravity measurements. In *Computers and Graphics (Pergamon)*. Elsevier, pp. 866–883.

Kurz, D. & Ben Himane, S., 2011. Inertial sensor-aligned visual feature descriptors. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp. 161–166.

Lagendijk, R.L. & Biemond, J., 1990. *Iterative Identification and Restoration of Images*,

Lang, P. et al., 2002. Inertial tracking for mobile augmented reality. In *IMTC/2002. Proceedings of the 19th IEEE Instrumentation and Measurement Technology Conference (IEEE Cat. No.00CH37276)*. IEEE, pp. 1583–1587.

Lee, G.H., Fraundorfer, F. & Pollefeys, M., 2011. RS-SLAM: RANSAC sampling for visual FastSLAM. *IEEE International Conference on Intelligent Robots and Systems*, pp.1655–1660.

Lefferts, E.J., Markley, F.L. & Shuster, M.D., 1982. Kalman Filtering for Spacecraft Attitude Estimation. *Journal of Guidance, Control, and Dynamics*, 5(5), pp.417–429.

Lepetit, V., Moreno-Noguer, F. & Fua, P., 2008. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision*, 81(2), pp.155–166.

Leutenegger, S., Chli, M. & Siegwart, R.Y., 2011. BRISK: Binary Robust invariant scalable keypoints. *2011 International Conference on Computer Vision*, pp.2548–2555.

Lhuillier, M., 2005. Automatic Structure and Motion using a Catadioptric Camera. *Proceedings of the 6th Workshop on Omnidirectional Vision, Camera Networks and Non-Classical Cameras*.

Li, M., Kim, B.H. & Mourikis, A.I., 2013. Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera. *Proceedings - IEEE International Conference on Robotics and Automation*, pp.4712–4719.

Li, M. & Mourikis, A.I., 2013. High-precision, consistent EKF-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6), pp.690–711.

Li, M. & Mourikis, A.I., 2012. Improving the accuracy of EKF-based visual-inertial odometry. In *2012 IEEE International Conference on Robotics and Automation*. IEEE, pp. 828–835.

Li, R., Zeng, B. & Liou, M.L., 1994. New three-step search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(4), pp.438–442.

Lindeberg, T., 1998. Feature Detection with Automatic Scale Selection. *International Journal of Computer Vision*, 30(2), pp.79 – 116.

Liu, F. et al., 2009. Content-preserving warps for 3D video stabilization. *ACM Transactions on Graphics*, 28, p.1.

Lobo, J. & Dias, J., 2003. Vision and inertial sensor cooperation using gravity as a vertical reference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12), pp.1597–1608.

Longuet-Higgins, H.C., 1981. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828), pp.133–135.

Lowe, D.G., 2004. Distinctive Image Features from Scale-invariant Keypoints. , pp.1–28.

Lucas, B. & Kanade, T., 1981. An iterative image registration technique with an application to stereo vision. In *IJCAI'81 Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*.

Lui, J.S. & Chen, R., 1998. Sequential Monte Carlo Methods for Dynamic Systems. *Journal of the American Statistical Association*, 93(443), pp.1032–1044.

Mahony, R., Hamel, T. & Pflimlin, J., 2008. Non-linear complementary filters on the special orthogonal group. , XX(Xx), pp.1203–1217.

Mahony, R., Hamel, T. & Pflimlin, J.-M., 2005. Complementary filter design on the special orthogonal group SO(3). *Proceedings of the 44th IEEE Conference on Decision and Control*, (1), pp.1477–1484.

Mair, E. et al., 2010. Adaptive and generic corner detection based on the accelerated segment test. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6312 LNCS, pp.183–196.

Marins, J.L. et al., 2003. An extended Kalman filter for quaternion-based orientation estimation using MARG sensors. *Proceedings 2001 IEEE/RSJ International Conference*

Appendix D: References

on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180), 4, pp.2003–2011.

Marquardt, D.W., 1963. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2), pp.431–441.

Martin, P. et al., 2014. Mapping and re-localization for mobile augmented reality. In *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, pp. 3352–3356.

Matas, J. et al., 2004. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10), pp.761–767.

Matas, J. & Chum, O., 2005. Randomized RANSAC with sequential probability ratio test. In *Proceedings of the IEEE International Conference on Computer Vision*. pp. 1727–1732.

Matas, J. & Chum, O., 2004. Randomized RANSAC with Td,d test. *Image and Vision Computing*, 22(10), pp.837–842.

Mikolajczyk, K. et al., 2005. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2), pp.43–72.

Mikolajczyk, K., 2004. Scale & Affine Invariant Interest Point Detectors. *International Journal of Computer Vision*, 60(1), pp.63–86.

Mikolajczyk, K. & Schmid, C., 2005. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27, pp.1615–1630.

Mikolajczyk, K. & Schmid, C., 2002. An affine invariant interest point detector. *Computer Vision - ECCV 2002*, 2350, pp.128–142.

Mikolajczyk, K. & Schmid, C., 2001. Indexing based on scale invariant interest points. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. IEEE Comput. Soc, pp. 525–531.

Milford, M.J., Wyeth, G.F. & Prasser, D., 2004. RatSLAM: a hippocampal model for simultaneous localization and mapping. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 1, pp.403–408 Vol.1.

Montemerlo, M., 2003. FastSLAM : A Factored Solution to the Simultaneous Localization and Mapping Problem With Unknown Data Association. *Techniques*, (July), p.123.

Moravec, H.P., 1980. Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover.

Moreels, P. & Perona, P., 2007. Evaluation of features detectors and descriptors based on 3D objects. *International Journal of Computer Vision*, 73(3), pp.263–284.

Moreno-Noguer, F., Lepetit, V. & Fua, P., 2007. Accurate Non-Iterative O(n) Solution to the PnP Problem. In *2007 IEEE 11th International Conference on Computer Vision*. IEEE, pp. 1–8.

Mouragnon, E. et al., 2006. Real Time Localization and 3D Reconstruction. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. IEEE, pp. 363–370.

Mourikis, A.I. & Roumeliotis, S.I., 2007. A multi-state constraint Kalman filter for vision-aided inertial navigation. In *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, pp. 3565–3572.

Moutarde, F., Stanciulescu, B. & Breheret, A., 2008. Real-time visual detection of vehicles and pedestrians with new efficient adaBoost features. *2nd Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV), at 2008 IEEE International Conference on Intelligent RObots Systems (IROS 2008)*.

Murray, R.M., Li, Z. & Sastry, S.S., 1994. *A Mathematical Introduction to Robotic Manipulation*,

Nistér, D., 2004. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6), pp.756–77.

Nistér, D., 2005. Preemptive RANSAC for live structure and motion estimation. *Machine Vision and Applications*, (Iccv), pp.1–29.

Nistér, D., Naroditsky, O. & Bergen, J., 2006. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23, pp.3–20.

Nistér, D. & Stewénius, H., 2006. A Minimal Solution to the Generalised 3-Point Pose Problem. *Journal of Mathematical Imaging and Vision*, 27(1), pp.67–79.

Panahandeh, G., Zachariah, D. & Jansson, M., 2012. Exploiting ground plane constraints for visual-inertial navigation. In *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*. IEEE, pp. 527–534.

Paz, L.M. et al., 2007. EKF SLAM updates in O(n) with Divide and Conquer SLAM. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, pp. 1657–1663.

Po, L. & Ma, W., 1996. A novel four-step search algorithm for fast block motion estimation. *Circuits and Systems for Video Technology, ...*, 6(June), pp.313–317.

Porzi, L. et al., 2012. Visual-inertial tracking on Android for Augmented Reality applications. In *2012 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems, EESMS 2012 - Proceedings*. pp. 35–41.

Raguram, R., Frahm, J.M. & Pollefeys, M., 2008. A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus. In *Lecture Notes in*

Appendix D: References

Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 500–513.

Rosten, E. & Drummond, T., 2006. Machine learning for high-speed corner detection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS, pp.430–443.

Rosten, E., Porter, R. & Drummond, T., 2010. Faster and better: A machine learning approach to corner detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pp.1–35.

Roumeliotis, S.I., Johnson, a. E. & Montgomery, J.F., 2002. Augmenting inertial navigation with image-based motion estimation. *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, 4(May), pp.4326–4333.

Rublee, E. et al., 2011. ORB: An efficient alternative to SIFT or SURF. *2011 International Conference on Computer Vision*, pp.2564–2571.

Sabatini, A.M., 2006. Quaternion-based extended Kalman filter for determining orientation by inertial and magnetic sensing. *IEEE transactions on bio-medical engineering*, 53(7), pp.1346–56.

Scaramuzza, D. & Fraundorfer, F., 2012. Tutorial: Visual odometry. *IEEE Robotics and Automation Magazine*, 18(June), pp.80–92.

Schmid, C. & Mohr, R., 1997. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5), pp.530–535.

Schöps, T., Engel, J. & Cremers, D., 2014. Semi-dense visual odometry for AR on a smartphone. In *IEEE International Symposium on Mixed and Augmented Reality*. pp. 1–6.

Shoemake, K., 1985. Animating rotation with quaternion curves. *ACM SIGGRAPH Computer Graphics*, 19(3), pp.245–254.

Solà, J. et al., 2011. Impact of Landmark Parametrization on Monocular EKF-SLAM with Points and Lines. *International Journal of Computer Vision*, 97(3), pp.339–368.

Stanciulescu, B., Breheret, A. & Moutarde, F., 2009. Introducing New AdaBoost Features for Real-Time Vehicle Detection.

Strasdat, H., Montiel, J.M.M. & Davison, A.J., 2010. Real-time monocular SLAM: Why filter? *2010 IEEE International Conference on Robotics and Automation*, pp.2657–2664.

Strelow, D. & Singh, S., 2003. Online Motion Estimation from Image and Inertial Measurements. *Proceedings of the Workshop on Integration of Vision and Inertial Sensors (INERVIS 2003)*.

Thrun, S., 2002. Probabilistic robotics. *Communications of the ACM*, 45(3), pp.52–57.

Tola, E. et al., 2010. DAISY an efficient dense descriptor applied to wide-baseline stereo.pdf. , pp.815–830.

Tordoff, B. & Murray, D., 2002. Guided sampling and consensus for motion estimation. *Lecture Notes in Computer Science*, (May 2002).

Torr, P.H.S. & Zisserman, A., 2000. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, 78(c), pp.138–156.

Triggs, B. et al., 2000. Bundle Adjustment—A Modern Synthesis. *Vision algorithms: theory and practice. S*, 1883, pp.298–372.

Varadarajan, V.S., 1974. *Lie Groups, Lie Algebras, and Their Representation*,

Wendel, A. et al., 2012. Dense reconstruction on-the-fly. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1450–1457.

Wheeler, M.D. & Ikeuchi, K., 1995. *Iterative Estimation of Rotation and Translation using the Quaternion*,

Yap, T., Mourikis, A.I. & Shelton, C.R., 2011. A particle filter for monocular vision-aided odometry. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, pp. 5663–5669.

You, S. & Neumann, U., 2001. Fusion of vision and gyro tracking for robust augmented reality registration. In *Proceedings IEEE Virtual Reality 2001*. IEEE Comput. Soc, pp. 71–78.

You, S., Neumann, U. & Azuma, R., 1999. Hybrid inertial and vision tracking for augmented reality registration. *Proceedings IEEE Virtual Reality (Cat. No. 99CB36316)*.

You, S., Neumann, U. & Azuma, R., 1999. Orientation tracking for outdoor augmented reality registration. *IEEE Computer Graphics and Applications*, 19(6), pp.36–42.

Zaklouta, F. & Stanciulescu, B., 2011. Real-time traffic sign recognition using spatially weighted HOG trees. In *2011 15th International Conference on Advanced Robotics (ICAR)*. IEEE, pp. 61–66.

Zhang, Z., 2000. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11), pp.1330–1334.

Zhu, S. & Ma, K.K., 2000. A new diamond search algorithm for fast block-matching motion estimation. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 9(3), p.525.

Zhu, Z. et al., 1998. Camera Stabilization Based on 2.5D Motion Estimation and Inertial Motion Filtering. *IEEE Int. Conf. on Intelligent ...*, pp.329–334.

Fusion de données capteurs étendue pour applications vidéo embarquées

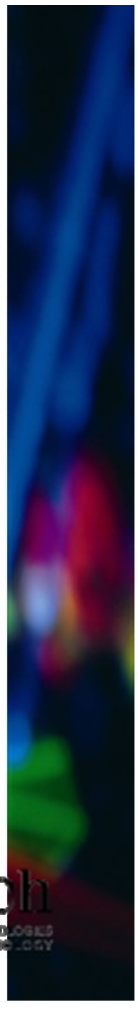
RESUME : Le travail réalisé au cours de cette thèse se concentre sur la fusion des données d'une caméra et de capteurs inertiels afin d'effectuer une estimation robuste de mouvement pour des applications vidéos embarquées. Les appareils visés sont principalement les téléphones intelligents et les tablettes. On propose une nouvelle technique d'estimation de mouvement 2D temps réel, qui combine les mesures visuelles et inertielles. L'approche introduite se base sur le RANSAC préemptif, en l'étendant via l'ajout de capteurs inertiels. L'évaluation des modèles de mouvement se fait selon un score hybride, un lagrangien dynamique permettant une adaptation à différentes conditions et types de mouvements. Ces améliorations sont effectuées à faible cout, afin de permettre une implémentation sur plateforme embarquée. L'approche est comparée aux méthodes visuelles et inertielles. Une nouvelle méthode d'odométrie visuelle-inertielle temps réelle est présentée. L'interaction entre les données visuelles et inertielles est maximisée en effectuant la fusion dans de multiples étapes de l'algorithme. A travers des tests conduits sur des séquences acquises avec la vérité terrain, nous montrons que notre approche produit des résultats supérieurs aux techniques classiques de l'état de l'art.

Mots clés : Estimation de mouvement, vision par ordinateur, capteurs inertiels, fusion capteur, temps-réel, embarqué, RANSAC, SLAM, odométrie, filtre à particule.

Extended sensor fusion for embedded video applications

ABSTRACT : This thesis deals with sensor fusion between camera and inertial sensors measurements in order to provide a robust motion estimation algorithm for embedded video applications. The targeted platforms are mainly smartphones and tablets. We present a real-time, 2D online camera motion estimation algorithm combining inertial and visual measurements. The proposed algorithm extends the preemptive RANSAC motion estimation procedure with inertial sensors data, introducing a dynamic lagrangian hybrid scoring of the motion models, to make the approach adaptive to various image and motion contents. All these improvements are made with little computational cost, keeping the complexity of the algorithm low enough for embedded platforms. The approach is compared with pure inertial and pure visual procedures. A novel approach to real-time hybrid monocular visual-inertial odometry for embedded platforms is introduced. The interaction between vision and inertial sensors is maximized by performing fusion at multiple levels of the algorithm. Through tests conducted on sequences with ground-truth data specifically acquired, we show that our method outperforms classical hybrid techniques in ego-motion estimation.

Keywords : Motion estimation, computer vision, inertial sensors, sensor fusion, real-time, embedded, RANSAC, SLAM, odometry, particle filter.



MINES ParisTech

ParisTech